Constrained Niching Using Differential Evolution

D. J. Poole[1], C. B. Allen[2]

Department of Aerospace Engineering, Queens Building, University Walk, University of Bristol, Bristol, BS8 1TR, U.K.

Abstract

The work presented here involves development and detailed investigations of niching methods for multimodal optimization of constrained functions. There is a lack of investigations in the literature on constrained multimodal optimization, hence a number of constrained niching algorithms have been developed here that leverage existing differential evolution-based niching methods with a feasibility rules-domination selection procedure. Furthermore, a suite of 18 benchmark functions has been developed and are presented in this paper; nine newly developed functions are incorporated with nine existing functions from the literature. Optimization results on these analytical functions using the constrained niching algorithms are presented, with analysis provided on the ability to locate multiple global optima, the convergence speed and constraint handling nature of the methods. The differential evolution strategy is also investigated, with SHADE and L-SHADE strategies considered. Finally, a dimensionality study also compares against the only other known constrained niching algorithm. Results indicate that all of the algorithms developed and tested generally perform well for low dimensional, low modality problems, but that local neighbourhood-based methods show the best results across the suite of functions tested. When high-dimensional problems are considered, using the L-SHADE strategy yields excellent results. An accompanying supplementary data file is provided with the manuscript.

Keywords: Multimodal Optimization, Niching, Constraints, Differential Evolution

1. Introduction

In engineering, optimization involves changing an aspect of design to improve some cost measure. In real-world design problems, however, constraints that represent barriers to solutions exist, and the final optimized solution must be feasible i.e. all of the constraints are satisfied. These types of problems are termed constrained numerical optimization problems (CNOPs), which are being solved using computationally expensive simulations for cost evaluations in, for example, the aerospace industry [1–5].

Nature-inspired meta-heuristics have become commonplace when solving traditional unconstrained numerical optimization problems (UNOPs). Typically, these approaches use a population of individuals who cooperate together in search of the optimal solution; evolutionary algorithms (EAs)–such as genetic algorithms (GAs) [6]–and swarm intelligence algorithms (SIAs)–such as particle swarm optimization (PSO)–are popular nature-inspired approaches. Differential evolution (DE) [7] is a specific example of an EA, which is simple, yet has proved effective in many areas of numerical optimization [8]. DE is a population-based optimization algorithm, where operators for mutation, crossover and selection act to create a new population from an existing one such that eventually, through a series of iterations, the population will converge on to a globally optimal solution.

When solving a CNOP, nature-inspired algorithms typically have to be coupled with a constraint handling method, such that a feasible solution can be found. Constraint handling methods commonly either modify the objective being solved (such as penalty functions), or modify the underlying mechanics of the search algorithm to make it suitable for

Email addresses: d.j.poole@bristol.ac.uk (D. J. Poole), c.b.allen@bristol.ac.uk (C. B. Allen)

[1]Lecturer. Corresponding author.

[2]Professor of Computational Aerodynamics.

solving CNOPs. A variety of constraint handling approaches have been coupled to nature-inspired algorithms, and been shown to be successful; see [9–16], for example.

Whether solving a UNOP or a CNOP, generally the goal is to locate the overall best (feasible) solution within the search space. While in some cases, locating the most optimal solution is the ideal result, in other problems it may be preferable to locate all of the optimal solutions (whether these are global or local), which is called multimodal optimization, and this is the focus of the current work. To solve multimodal optimization problems, various techniques called "niching" methods have been developed; see [17–22], for example. Often, the underlying goal of these methods is to enhance diversity in the population such that optimal solutions that lie at different parts of the search space are located. The primary issue surrounding the majority of niching methods is one of complexity; the number of calculations tends to be of $O(N^2)$ (where N is the number of individuals in the population), whereas the complexity of the underlying SIA or EA is typically $O(N)$ [23].

There has been a rich history of research in both constraint handling and multimodal optimization using nature-inspired algorithms. For example, a comprehensive review on constraint handling has been presented by Mezura-Montes and Coello Coello [24], and on multimodal optimization by Li *et al.* [25]. Furthermore, at the IEEE Congress on Evolutionary Computation, since 2005 there have been three competitions (where results from algorithms solved on a number of benchmark functions are compared and ranked based on their performance) on constraint handling using nature-inspired algorithms and three competitions on multimodal optimization[3]. A separate consortium have also organised three further competitions on multimodal optimization[4]. Despite the substantial research in both of these fields independently, there is little investigation on solving multimodal optimization of constrained optimization problems. In a recent study on niching methods [25], it was stated that "*there lacks a systematic study on how existing niching methods, largely designed for unconstrained optimization, should cope with constraints*".

The only theoretical contribution in this field to date comes from Deb and Saha [26, 27], who proposed an analytical function generator for the user to produce functions with a specified number of dimensions and constraints, which was then solved using a modified version of the Non-dominated Sorting Genetic Algorithm II (NSGA-II) [28]. Despite the lack of theoretical research into constrained multimodal optimization, there exists an appetite for such methods, particularly in engineering optimization. For example, a recently-proposed shape optimization problem requires finding all the optima of an aircraft wing optimization problem subject to a substantial number of linear and non-linear constraints [29]. Furthermore, other constrained engineering optimization problems exhibit a strong multimodal tendency, for example in truss design [30], and as such, studies and methods for solving such problems are required.

This paper therefore considers the problem of solving constrained multimodal optimization problems. Due to the shortage of research in this field, there is a lack of comprehensive test cases on which to test potential algorithms, so a suite of new constrained multimodal analytical test functions have been developed and are presented that contain a mixture of functions with multiple global optima, and ones with multiple global and local optima. Various constrained niching DE algorithms that leverage existing unconstrained niching DE algorithms with a feasibility-rules approach for handling constraints are presented, which are then used to solve the suite of constrained multimodal analytical functions. Finally, more recent DE variants are introduced into the constrained niching algorithms to investigate the effect of using a different DE-based algorithm as a basis to construct constrained niching methods. A number of the highest performing algorithms are also compared to the modified NSGA-II algorithm presented by Deb and Saha [26, 27], which to the authors' knowledge, is the only other constrained niching algorithm to have been published; a dimensionality study is presented for this.

The remainder of this paper is organised as follows: section 2 poses the multimodal problem and gives a background review on constraint handling and niching algorithms; section 3 provides an outline of DE; section 4 gives details on the new suite of low-dimensional analytical constrained multimodal functions developed as part of this work; section 5 outlines the new canonical-DE constrained niching algorithms and provides results of a parameter tuning process; section 6 provides the results and discussion of the new algorithms' performance on the low-dimensional function suite; section 7 presents results when more recent DE variants are used; section 8 provides a dimensions and constraint study comparing the best performing constrained DE niching algorithms against a niching form of NSGA-

[3] See the pages of P. Suganthan for more information: `www3.ntu.edu.sg/home/epnsugan/index_files/`

[4] See the pages of M. Epitropakis for more information: `www.epitropakis.co.uk/`

II; finally, section 9 provides concluding remarks and notes on future research directions. Throughout the paper, figure and table labels that are denoted by 'S' may be found in the supplementary material that accompany the paper.

2. Background

In this section, a background review of constraint handling and multimodal optimization in unconstrained search spaces is presented. Finally, the multimodal solution to a CNOP is given. The general definition of a CNOP is described in equation 1.

$$\begin{aligned} \min_{\mathbf{x}\in\mathcal{S}\in\Re^D} \quad & f(\mathbf{x}) \\ \text{subject to} \quad & g_i(\mathbf{x}) \leq 0 \quad , \quad i \in \{1,\ldots,p\} \\ & h_j(\mathbf{x}) = 0 \quad , \quad j \in \{1,\ldots,q\} \end{aligned} \tag{1}$$

In equation 1, $\mathbf{x}$ is the solution vector $[x^1, x^2, \ldots, x^D]^T$ where each element of the vector is a design variable in a problem containing D design variables; $f(\mathbf{x})$ is the value of the objective function for the given solution vector; $g_i(\mathbf{x})$ represents the i-th inequality constraint of a total of p inequality constraints; $h_j(\mathbf{x})$ represents the j-th equality constraint of a total of q inequality constraints where $p+q$ is the total number of constraints; $\mathcal{S}$ is the bounded region of $\Re^D$ where the solution exists, called the design space, which is a D-orthotope defined by an upper bound in the d-th dimension, U^d, and a lower bound, L^d, where $d \in \{1,\ldots,D\}$. The feasible region, $\mathcal{F}$, defines the set of all feasible solutions.

The majority of nature-inspired optimization algorithms are unable to solve CNOPs directly so constraint handling methods have to be employed. A short review is provided here on constraint handling using nature-inspired algorithms. A comprehensive review of constraint handling in nature inspired numerical optimization algorithms has been performed by Mezura-Montes and Coello Coello [24], as well as in the book by Datta and Deb [31].

An often simple, and typically classic way to handle constraints is to amalgamate the objective function and constraints of a CNOP to transform it into an UNOP via a penalty function, on which, traditional nature-inspired optimization algorithms can be applied. A naive implementation of a penalty is to simply eliminate any infeasible individuals from the search. Termed the 'death-penalty' , it is reasonably simple to implement though is an inefficient approach [32, 33]. Penalty functions which are simply a weighted summation of the constraint violation, such as that presented for PSO by Venter and Sobieszczanski-Sobieski [9], can also be achieved. However, more effective approaches have looked at dynamically changing the penalty function depending on the current and past state of the search, which are called adaptive penalties. A review of adaptive penalties has been presented by Barbosa *et al.* [34]. Adaptive penalties have the advantage of automatically changing the penalty function to adapt to the current search, thus reducing (or even eliminating) any user required input of penalty parameters. These can use functions of the current objective and constraints such as in [35] or [10], or use co-evolution [11], where the penalty function is evolved along with the problem using a EA.

The antithesis to the idea of penalty functions is, instead of amalgamating the objective and constraints, keeping the objective and constraint problems separate. Results using this separation idea are positive, demonstrating at least as good results as other penalty approaches [12, 13, 36] . A slight modification of the separation approach is by the feasibility rules approach (also called binary tournament selection, first proposed by Deb [14]) which makes a decision about which of two locations 'win' a binary tournament based on feasibility or constraint violation [15, 37]. Similar to the idea of the feasibility rules are the α-constrained [38] and ϵ-constrained [39, 40] methods, where a satisfaction (or tolerance) represents the relaxation allowed. A further extension to this idea is to treat the CNOP as a bi-objective problem, where the objectives are to both solve the objective function and solve the constraints [41, 42].

Further, special operators that alter specific mechanics of nature-inspired algorithms have also been popular. For example, when using a SIA, the idea of feasible directions [43], which is a direction that reduces the value of the objective function while pointing towards the feasible space, is popular [9, 16, 44]. Other, more sophisticated approaches have also been shown to perform well, such as that presented by Lu and Chen [45], who designed an approach which adapts a search based on the size and topology characteristics of the search space.

Specifically concerning DE, much work has been presented that modifies various aspects of DE to produce algorithms that can explicitly handle constraints, see [46–50] for examples. An ensemble of constraint handling methods

has also been successfully applied to DE [51]. The ensemble approach takes any number of constraint handling techniques and adaptively applies them to exploit the individual performance of each technique. Mezura-Montes *et al.* [52] investigated the effect of different mutation strategies in DE on constrained optimization. They highlighted the fact that much work in solving CNOPs using DE uses the feasibility rules of Deb[14], and as such, used this to handle the constraints.

Commonly, population diversity is a requirement of EAs, such that design space exploration is achieved, with diversity decreasing through the optimization to exploit the global optimum within the space. On the other hand, while diversity is important for unimodal optimization, this is not necessarily the main driver of niching techniques. Niching requires a more careful balance of diversity to be able to locate and maintain strong and stable niches, and therefore exploit all the optima in the design space. Comprehensive reviews on niching techniques have been provided by Das *et al.* [53] and more recently by Li *et al.*[25] (the reader is particularly guided to the review of Li *et al.* for discussions of niching using SIAs as well as EAs), and a short review is presented here with particular attention paid to niching using EAs.

One of the earliest bodies of work on maintaining population diversity, and therefore implicitly also multimodal optimization, was by de Jong [17], who introduced the classical idea of crowding. Crowding compares the fitness of close individuals and attempts to promote a population that is spread widely but has good fitness. Mahfoud [54] later suggested the deterministic crowding algorithm to improve on the basic crowding idea given by de Jong. Thomsen [18] was the first to couple crowding with DE to produce the crowding-DE (CDE) algorithm. The basic crowding method, and therefore also CDE, requires specification of the crowding factor, which determines the size of the subset of individuals used to determine the closest individual, although this is normally set to be the population size [18].

Another classic niching method is fitness sharing [6, 19], which penalises crowding together of individuals. Fitness sharing works by penalising the fitness of those individuals who all fall within a niching radius, σ. The setting of the niching radius can often be complex and specific to a certain problem. An incorrect setting of σ can lead to poor performance of the algorithm [55]. Thomsen[18] was the first to suggest the fitness sharing DE, which used the classic sharing method, but that the population was increased into a super-population of double the size, from which the selection using fitness sharing was performed. When comparing their two new niching DE algorithms, Thomsen determined that CDE outperformed sharing DE.

Two further methods that both require setting of a radius parameter are clearing [20] and speciation [21, 56]. Clearing works by removing poor individuals who lie within the niching radius, while speciation creates species who lie within a niching radius. The evolution then occurs within the species. This idea of local populations is something that is popular in niching algorithms. For example, Qu *et al.* [22] developed a neighbourhood framework that was used to present new, neighbourhood-based crowding, speciation and fitness sharing DE algorithms. These were shown to outperform their non-neighbourhood counterparts.

Other DE niching methods that have shown to perform well include using a dynamic archive [57], probabilistic selection [58] and local neighbourhoods [59, 60]. Furthermore, an ensemble of niching methods, with dynamic selection to determine which to use, has also been successful [61].

The body of work presented above on constrained optimization is solving the problem of finding the unique solution that solves a CNOP. However, the work presented in this paper extends this idea to solving a multimodal CNOP using constrained niching algorithms. In this work, it is only the location of all global optima that are considered. The multimodal solution of a CNOP is therefore the set containing N_g global optima, $\mathcal{X}^* \subseteq \mathcal{F}$ where $\mathcal{X}^* = \{\mathbf{x}^{*1}, \ldots, \mathbf{x}^{*N_g}\}$ that minimise f in $\mathcal{F}$, hence:

$$f(\mathbf{x}^{*1}) = \cdots = f(\mathbf{x}^{*N_g}) < f(\mathbf{x}) \, , \forall \mathbf{x} \mid \mathbf{x} \in \mathcal{F}, \mathbf{x} \notin \mathcal{X}^*$$

3. Differential Evolution

From the review presented above, it is clear that much work has been done that uses DE for solving CNOPs as well as solving multimodal optimization problems. As such, DE is used as a basis for the development of the constrained niching algorithms presented later in this paper. DE was first presented by Storn and Price [7, 62] and has since been developed into a widely used global search algorithm. A full review on the development and applications of DE is outside the scope of this paper, however, in-depth reviews have been presented by Das and Suganthan [63], Neri and Tirronen [64] and more recently by Das *et al.* [8].

DE uses a population of individuals who evolve through the iterations of the optimization. The total population, $\mathcal{X}$, is composed of N individuals, where the n-th individual is represented by a target vector that details an individual's position in the design space, hence $\mathcal{X} = [\mathbf{x}_1, \ldots, \mathbf{x}_N]^T$. The nomenclature used in this paper to represent the target vector of the n-th individual at the t-th iteration of the optimization, is given by:

$$\mathbf{x}_n(t) = [x_n^1(t), x_n^2(t), \ldots, x_n^D(t)]^T \tag{2}$$

In this paper, the canonical DE algorithm (that given by Storn and Price [7, 62]) is used as well as more recent DE algorithms. The more recent DE algorithms are success-history based adaptive DE (SHADE) [65] and SHADE with linear population size reduction (L-SHADE) [66]. Each of these are outlined below. The initial stages of the work presented in this paper use the canonical DE algorithm with the DE/rand/1 mutation strategy are used (see below for details), however, later on, a different mutation strategy (DE/best/1) is also considered as well as the more recent DE algorithms.

3.1. Canonical DE

The canonical DE algorithm follows five steps to advance the optimization algorithm, which are given as:

1. *Initialisation*:
   Within the design space $\mathcal{S}$, the initial target vectors of the N individuals are generated. This is commonly done randomly such that the target vector at the 0-th iteration (i.e. the initial location of the individual) in the d-th dimension ($d \in \{1, \ldots, D\}$) is given as:

   $$x_n^d(0) = L^d + \text{rand}(0,1)(U^d - L^d) \tag{3}$$

   where $\text{rand}(0,1)$ is a uniformly distributed random number on the interval $[0,1]$.
2. *Mutation*:
   For each individual within the population, a donor vector is generated using scaled differences between other individuals within the population. A number of mutation strategies have been proposed and two are investigated in this paper. The n-th donor vector, $\mathbf{v}_n$, using a DE/rand/1 mutation strategy, is given by:

   $$\mathbf{v}_n = \mathbf{x}_{r_1}(t) + F(\mathbf{x}_{r_2}(t) - \mathbf{x}_{r_3}(t)) \tag{4}$$

   where F is the scaling factor, and r_1, r_2 and r_3 are uniformly distributed random integers on the interval $[1, N]$ such that $r_1 \neq r_2 \neq r_3 \neq n$. The n-th donor vector using the DE/best/1 strategy is:

   $$\mathbf{v}_n = \mathbf{x}_b(t) + F(\mathbf{x}_{r_1}(t) - \mathbf{x}_{r_2}(t)) \tag{5}$$

   where:

   $$b = \underset{i \in \{1,\ldots,N\}}{\arg\min}\ f(\mathbf{x}_i)$$

   so $\mathbf{x}_b$ is the best individual in the population. Also, r_1 and r_2 are uniformly distributed random integers on the interval $[1, N]$ such that $r_1 \neq r_2 \neq b$ and $r_1 \neq r_2 \neq n$.
3. *Crossover*:
   To enhance diversity, crossover of the individual elements of the target vector and the donor vector is employed to produce a trial vector. Binomial crossover is used throughout this paper which produces the n-th trial vector in the d-th dimension, u_n^d, by the following equation:

   $$u_n^d = \begin{cases} v_n^d & \text{if } \text{rand}(0,1) \leq CR \text{ or } r_n = d \\ x_n^d(t) & \text{otherwise} \end{cases} \tag{6}$$

   where r_n is a uniformly distributed random integer on the interval $[1, D]$ and CR is the crossover constant. Hence, elements of the trial vector are accepted from the donor vector at a probability of CR, and at least one component of the donor vector is accepted.

4. *Selection*:
   The trial vector is tested and the n-th target vector at the next iteration is generated by the following relationship:

$$\mathbf{x}_n(t+1) = \begin{cases} \mathbf{u}_n & \text{if } f(\mathbf{u}_n) \leq f(\mathbf{x}_n(t)) \\ \mathbf{x}_n(t) & \text{otherwise} \end{cases} \tag{7}$$

   Hence, the target vector is replaced with the trial vector if the trial vector is atleast as good as the target vector.
5. *Stopping conditions*:
   Once all N target vectors have been updated, if the stopping conditions have been reached (which throughout this paper is whether the maximum allowed number of function evaluations have been performed), the algorithm exits. If not, then the processes of steps 2-5 are repeated.

A pseudo-code of the DE algorithm is given in Algorithm 1 where FEs_{max} is the maximum permitted number of function evaluations and FEs is the current number of function evaluations. An example of the mechanism of DE is given in figure 1, where the mutation mechanism (illustrated in red), is a scaled difference vector between two randomly sampled individuals applied to a further random individual. The binomial crossover mechanism is then illustrated in blue where elements of $\mathbf{v}_n$ are selected to produce a trial vector. In this example, $\widehat{\mathbf{u}_n}$, $\widetilde{\mathbf{u}_n}$ and $\mathbf{v}_n$ are the possible trial vectors that could be generated.

Algorithm 1 DE algorithm

```
Randomly initialise individuals and calculate objective
while FEs<FEs_max do
    for n = 1 → N do
        Perform mutation: equation 4 or 5
        Perform binomial crossover: equation 6
        Calculate objective and constraints of trial vector
    end for
    for n = 1 → N do
        Update n-th target vector: equation 7
    end for
end while
```

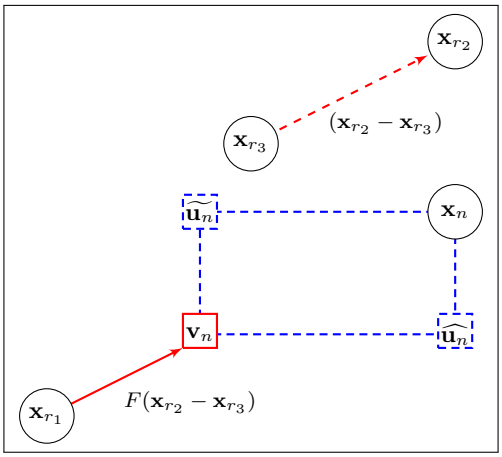


Figure 1: Illustration of DE/rand/1 mutation (red) and crossover (blue) in differential evolution in a two design variable space

3.2. SHADE

The SHADE algorithm, proposed by Tanabe and Fukunaga [65], is an extension on the canonical DE algorithm which uses adaptation of the F and CR parameters based on the history of these parameters through the search.

Furthermore, a mutation strategy called DE/current-to-$p$best/1 (which is a generalised form the the DE/current-to-best/1 strategy) is used as well as an optional external archive. The mutation strategy and archive are based on the JADE algorithm [67].

The SHADE algorithm retains the basic structure of the canonical DE algorithm with steps for initialisation, mutation, crossover, selection and termination, however the differences are highlighted below.

SHADE uses a historical memory for the CR and F parameters. The historical memory typically retains N historical parameters of CR and F, where the h-th entry in the CR history is $M_{CR,h}$ and in the F history is $M_{F,h}$. During the initialisation stage, all values in the history are set to 0.5. During the mutation stage, the scaling factor F is applied individual by individual. This approach is also used during mutation for the CR parameters. To generate these factors for the n-th individual, a random index r_n is generated on the interval $(1, N)$, then the following equations are applied:

$$CR_n = \text{randn}_n(M_{CR,r_n}, 0.1) \tag{8}$$

$$F_n = \text{randc}_n(M_{F,r_n}, 0.1) \tag{9}$$

where randn(μ, σ^2) and randc(μ, σ^2) are random values selected from a normal and Cauchy distribution respectively, with mean, μ, and variance, σ^2. If the value of CR_n is outside the interval $[0, 1]$, the limit value is used. Also, if $F_n > 1$, then $F_n = 1$, while if $F_n \leq 0$, F_n is regenerated until $F_n > 0$.

The mutation stage uses a DE/current-to-$p$best/1 strategy. The donor vector is generated according to:

$$\mathbf{v}_n = \mathbf{x}_n(t) + F_n(\mathbf{x}_{b_p}(t) - \mathbf{x}_n(t)) + F_n(\mathbf{x}_{r_1}(t) - \mathbf{x}_{r_2}(t)) \tag{10}$$

where $\mathbf{x}_{b_p}$ is a random individual selected from the p-best individuals in the population, where typically, p is set to be not larger than 0.2. For example, if $p = 0.2$, then $\mathbf{x}_{b_p}$ would be a randomly selected individual from the 20% best individuals in the whole population.

The crossover stage follows the same process as in the canonical DE to generate a trial vector. In the selection stage, the acceptance of the trial vector also follows the same process as the canonical DE algorithm, with the following addition. If the n-th trial vector, $\mathbf{u}_n$ is accepted, then F_n and CR_n are recorded in the set S_F and S_{CR} (this set is emptied at the start of each iteration).

At the end of an iteration, the historical memories are updated. For the h-th entry in the memory of CR, if there are any entries in the set S_{CR}, $M_{CR,h}$ is updated by a weighted mean:

$$M_{CR,h} = \sum_{i=1}^{|S_{CR}|} w_i S_{CR,i} \tag{11}$$

where:

$$w_i = \frac{|f(\mathbf{u}_i) - f(\mathbf{x}_i)|}{\sum_{j=1}^{|S_{CR}|} |f(\mathbf{u}_j) - f(\mathbf{x}_j)|} \tag{12}$$

while $M_{F,h}$ is updated by a weighted Lehmer mean:

$$M_{F,h} = \frac{\sum_{i=1}^{|S_F|} w_i S_{F,i}^2}{\sum_{i=1}^{|S_F|} w_i S_{F,i}} \tag{13}$$

where h is a parameter that increments every iteration that has at least one accepted trial vector (it resets to $h = 1$ if $h > N$).

A pseudo-code of the SHADE algorithm is given in Algorithm 2.

Algorithm 2 SHADE algorithm

Randomly initialise individuals and calculate objective
Set all values in M_{CR} and M_F to 0.5 and $h = 1$
while FEs<FEs$_{max}$ **do**
  $S_{CR} = \emptyset$ and $S_F = \emptyset$
  for $n = 1 \rightarrow N$ **do**
    Determine CR_n and F_n: equations 8 and 9
    Perform mutation: equation 10
    Perform binomial crossover: equation 6
    Calculate objective and constraints of trial vector
  end for
  for $n = 1 \rightarrow N$ **do**
    Update n-th target vector: equation 7
    If n-th target accepted, add CR_n and F_n to S_{CR} and S_F respectively
  end for
  If $S_{CR} \neq \emptyset$ and $S_F \neq \emptyset$, update $M_{CR,h}$ and $M_{F,h}$ (equations 11 and 13) and $h++$
end while

3.3. L-SHADE

SHADE with linear population size reduction (called L-SHADE), which is an improved version of SHADE, also presented by Tanabe and Fukunaga [66]. L-SHADE is implemented in the same way as SHADE, however, the population size is reduced through the iterations. At the start of the t-iteration, the population size is given by:

$$N(t) = \text{round}\left(\frac{N^{min} - N(0)}{\text{FEs}_{max}}\text{FEs} + N(0)\right) \tag{14}$$

where N^{min} is the minimum number of individuals permitted (typically this is determined by the mutation strategy; for example, DE/current-to-$p$best/1 requires at least 4 individuals) and $N(0)$ is the number of individuals at iteration 0. Whenever the population is reduced, the individuals are sorted and the $N(t)$-best individuals are kept in the population. L-SHADE also modifies the scheme for generating values of CR_n. During the update of the crossover memory, M_{CR}, the h-th value is assigned a 'terminal value', $\perp$, if all elements in S_{CR} are zero. The crossover index is then given by:

$$CR_n = \begin{cases} 0 & \text{if } M_{CR,r_n} = \perp \\ \text{randn}_n(M_{CR,r_n}, 0.1) & \text{otherwise} \end{cases} \tag{15}$$

Furthermore, the h-th entry in the crossover historical memory, $M_{CR,h}$, is updated by a weighted Lehmer mean instead of a weighted mean (as is done in SHADE):

$$M_{CR,h} = \frac{\sum_{i=1}^{|S_{CR}|} w_i S_{CR,i}^2}{\sum_{i=1}^{|S_{CR}|} w_i S_{CR,i}} \tag{16}$$

A pseudo-code of the L-SHADE algorithm is given in Algorithm 3.

4. Low-Dimensional Analytical Function Suite

As noted in the introduction, there has been little work on constrained multimodal optimization (this was also emphasised by Li *et al.* [25] in their review on multimodal optimization) and as such there is a lack of analytical test functions on which to test a constrained multimodal algorithm. Deb and Saha [26, 27] have presented a constrained multimodal test-problem generator where single-objective problems with a user-specified number of constraints can be generated that contain multiple global optima. Apart from this, there appears to be no other work that suggests constrained multimodal test problems. As such, a number of low-dimensional test problems have been developed and are presented here. The goal with these functions is to provide a number of low-dimensional, generally simple

Algorithm 3 L-SHADE algorithm

Randomly initialise individuals and calculate objective
Set all values in M_{CR} and M_F to 0.5 and $h = 1$
while FEs<FEs$_{max}$ **do**
  $S_{CR} = \emptyset$ and $S_F = \emptyset$
  Update N (equation 14) and keep N-best
  for $n = 1 \rightarrow N$ **do**
    Determine CR_n and F_n: equations 15 and 9
    Perform mutation: equation 10
    Perform binomial crossover: equation 6
    Calculate objective and constraints of trial vector
  end for
  for $n = 1 \rightarrow N$ **do**
    Update n-th target vector: equation 7
    If n-th target accepted, add CR_n and F_n to S_{CR} and S_F respectively
  end for
  If $S_{CR} \neq \emptyset$ and $S_F \neq \emptyset$, update $M_{CR,h}$ and $M_{F,h}$ (equations 11 and 16) and $h++$
end while

functions to compare a number of the algorithms on. However, it is recognised that higher-dimensional, harder multimodal functions may also be useful so later in the paper, a dimensionality study is presented using some higher dimensional functions generated using the problem generator of Deb and Saha [26, 27].

In total, a suite of 18 low-dimensional functions are used which contain various numbers of inequality constraints, global optima and local optima. Table 1 summarises the key parameters of the 18 functions. For all problems, all constraints are active at the optimum solutions. Functions `F1` to `F12` contain only global optima, while `F13` to `F18` also contain local optima. Functions `F1` to `F9` are constructed from the test-problem generator of Deb and Saha [26, 27], while functions `F10` to `F18` are constructed by taking already existing multimodal functions and adding various constraints to them. As such, for functions `F10` and `F13` to `F18`, the unconstrained global optima become infeasible and new optima are introduced. On the other hand, for functions `F11` and `F12`, the original global optima remain.

Table 2 details the locations of the optimal solutions of each of the functions, where the coefficient $A_{j,d}$ is given as:

$$A_{j,d} = \left\lfloor \frac{j + 2^d - 1}{2^{d-1}} \right\rfloor \tag{17}$$

Table 1: Properties of analytical functions

Func	D	p	N_g	$f(\mathbf{x}^*)$
`F1`	1	1	2	1.0
`F2`	2	1	2	1.0
`F3`	2	2	4	1.6
`F4`	5	1	2	1.0
`F5`	5	3	8	1.729843561973525
`F6`	5	5	32	$2.\overline{27}$
`F7`	10	1	2	1.0
`F8`	10	3	8	1.393589488353574
`F9`	10	5	32	1.826836992013024
`F10`	1	1	10	0.875
`F11`	2	4	4	1.0
`F12`	2	8	4	1.0
`F13`	1	1	2	$10 - 5\sqrt{2}$
`F14`	1	1	10	$10 - 5\sqrt{2}$
`F15`	2	1	8	$20 - 10\sqrt{2}$
`F16`	2	1	24	$20 - 10\sqrt{2}$
`F17`	3	1	16	$30 - 15\sqrt{2}$
`F18`	5	1	64	$50 - 25\sqrt{2}$

Table 2: Location of global optima of analytical functions

F1	$x^{*1} = 1, x^{*2} = -1$
F2	$\mathbf{x}^{*1} = [0,1]^T, \mathbf{x}^{*2} = [0,-1]^T$
F3	$\mathbf{x}^{*1} = [\sqrt{0.8}, \sqrt{0.8}]^T$ $\mathbf{x}^{*2} = [\sqrt{0.8}, -\sqrt{0.8}]^T$ $\mathbf{x}^{*3} = [-\sqrt{0.8}, \sqrt{0.8}]^T$ $\mathbf{x}^{*4} = [-\sqrt{0.8}, -\sqrt{0.8}]^T$
F4	$\mathbf{x}^{*1} = [0,0,0,0,1]^T$ $\mathbf{x}^{*2} = [0,0,0,0,-1]^T$
F5	$x_1^{*j} = (-1)^{A_{j,1}} 0.629371992938506$ $x_2^{*j} = (-1)^{A_{j,2}} 0.645108721639740$ $x_3^{*j} = 0, x_4^{*j} = 0$ $x_5^{*j} = (-1)^{A_{j,3}} 0.957898321192014$
F6	$x_d^{*j} = (-1)^{A_{j,d}} \sqrt{\overline{0.45}}$
F7	$\mathbf{x}^{*1} = [0,0,0,0,0,0,0,0,0,1]^T$ $\mathbf{x}^{*2} = [0,0,0,0,0,0,0,0,0,-1]^T$
F8	$x_1^{*j} = (-1)^{A_{j,1}} 0.442990013929914$ $x_2^{*j} = (-1)^{A_{j,2}} 0.455649419140285$ $x_3^{*j} = 0, x_4^{*j} = 0, x_5^{*j} = 0, x_6^{*j} = 0$ $x_7^{*j} = 0, x_8^{*j} = 0, x_9^{*j} = 0$ $x_{10}^{*j} = (-1)^{A_{j,3}} 0.994853226737024$
F9	$x_1^{*j} = (-1)^{A_{j,1}} 0.462013517275990$ $x_2^{*j} = (-1)^{A_{j,2}} 0.468626768691906$ $x_3^{*j} = (-1)^{A_{j,3}} 0.476441515992551$ $x_4^{*j} = (-1)^{A_{j,4}} 0.485653252797998$ $x_5^{*j} = 0, x_6^{*j} = 0, x_7^{*j} = 0$ $x_8^{*j} = 0, x_9^{*j} = 0$ $x_{10}^{*j} = (-1)^{A_{j,5}} 0.964838770685610$
F10	$x^{*j} = \frac{2j-1}{20}$
F11,F12	$\mathbf{x}^{*1} = [3,2]^T$ $\mathbf{x}^{*2} = [-2.805118, 3.131312]^T$ $\mathbf{x}^{*3} = [-3.779310, -3.283186]^T$ $\mathbf{x}^{*4} = [3.584428, -1.848126]^T$
F13	$x^{*1} = 0.375, x^{*2} = 0.625$
F14	$x^{*1} = 0.075, x^{*2} = 0.125$ $x^{*3} = 0.275, x^{*4} = 0.325$ $x^{*5} = 0.475, x^{*6} = 0.525$ $x^{*7} = 0.675, x^{*8} = 0.725$ $x^{*9} = 0.875, x^{*10} = 0.925$
F15	$\mathbf{x}^{*1} = [0.375, 0.1875]^T, \mathbf{x}^{*2} = [0.375, 0.3125]^T$ $\mathbf{x}^{*3} = [0.375, 0.6875]^T, \mathbf{x}^{*4} = [0.375, 0.8125]^T$ $\mathbf{x}^{*5} = [0.625, 0.1875]^T, \mathbf{x}^{*6} = [0.625, 0.3125]^T$ $\mathbf{x}^{*7} = [0.625, 0.6875]^T, \mathbf{x}^{*8} = [0.625, 0.8125]^T$
F16	$x_1^{*j} = 0.1875\,, \quad j \in \{1,\ldots,6\}$ $x_1^{*j} = 0.3125\,, \quad j \in \{7,\ldots,12\}$ $x_1^{*j} = 0.6875\,, \quad j \in \{13,\ldots,18\}$ $x_1^{*j} = 0.8125\,, \quad j \in \{19,\ldots,24\}$ $x_2^{*j} = 0.125\,, \quad j \in \{1,7,13,19\}$ $x_2^{*j} = 5/24\,, \quad j \in \{2,8,14,20\}$ $x_2^{*j} = 11/24\,, \quad j \in \{3,9,15,21\}$ $x_2^{*j} = 13/24\,, \quad j \in \{4,10,16,22\}$ $x_2^{*j} = 19/24\,, \quad j \in \{5,11,17,23\}$ $x_2^{*j} = 0.875\,, \quad j \in \{6,12,18,24\}$
F17	$x_1^{*j} = 0.5 + (-1)^{A_{j,1}} 0.125$ $x_2^{*j} = 0.5 + (-1)^{A_{j,2}} 0.125$ $x_3^{*j} = 0.1875\,, \quad j \in \{1,5,9,13\}$ $x_3^{*j} = 0.3125\,, \quad j \in \{2,6,10,14\}$ $x_3^{*j} = 0.6875\,, \quad j \in \{3,7,11,15\}$ $x_3^{*j} = 0.8125\,, \quad j \in \{4,8,12,16\}$
F18	$x_1^{*j} = 0.5 + (-1)^{A_{j,1}} 0.125$ $x_2^{*j} = 0.5 + (-1)^{A_{j,2}} 0.125$ $x_3^{*j} = 0.5 + (-1)^{A_{j,3}} 0.125$ $x_4^{*j} = 0.5 + (-1)^{A_{j,4}} 0.125$ $x_5^{*j} = 0.1875\,, \quad j \in \{1,5,\ldots,61\}$ $x_5^{*j} = 0.3125\,, \quad j \in \{2,6,\ldots,62\}$ $x_5^{*j} = 0.6875\,, \quad j \in \{3,7,\ldots,63\}$ $x_5^{*j} = 0.8125\,, \quad j \in \{4,8,\ldots,64\}$

The precise formulations of the functions are detailed below.

4.1. F1 to F9

Functions F1 to F9 are generated using the constrained multi-modal problem (CMMP) definitions of Deb and Saha [26, 27]. The objective function to be minimised is:

$$f(\mathbf{x}) = \sum_{d=1}^{D} x_d^2$$

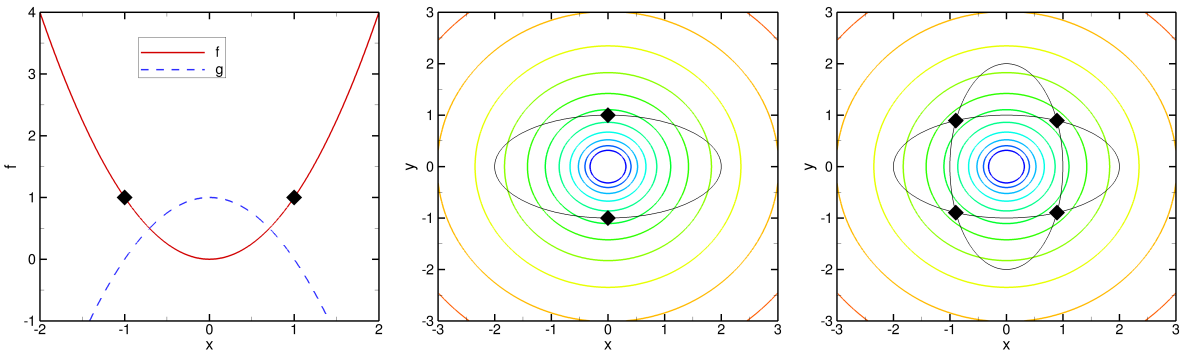
subject to:

$$\begin{aligned} g_1(\mathbf{x}) &= D^2 - \left(x_1^2 + 4x_2^2 + \cdots + D^2 x_D^2\right) \leq 0 \\ g_2(\mathbf{x}) &= D^2 - \left(D^2 x_1^2 + x_2^2 + \cdots + (D-1)^2 x_D^2\right) \leq 0 \\ &\vdots \\ g_p(\mathbf{x}) &= D^2 - \left(C_{p,1}^2 x_1^2 + C_{p,2}^2 x_2^2 + \cdots + C_{p,D}^2 x_D^2\right) \leq 0 \end{aligned}$$

where the bounds are $-(D+1) \leq x_d \leq (D+1)$ $(d \in \{1, \ldots, D\})$ and:

$$C_{p,i} = \begin{cases} (D-p+d+1) \bmod D & \text{if } (D-p+d+1) \bmod D \neq 0 \\ D & \text{otherwise} \end{cases}$$

The values of D and p for each of the nine functions are given in table 1. Plots of the uni-dimensional (F1) and bi-dimensional (F2 and F3) problems are given in figure 2, where for the bi-dimensional problems, the lines show the active constraint boundaries.



(a) F1 (b) F2 (c) F3

Figure 2: CMMP functions where black diamond symbols show location of optima

4.2. F10

Function F10 is a constrained version of the equal maxima function. The equal maxima function is univariate and has been used as part of the CEC 2013 multimodal benchmark suite [68], so for this paper, a cosine wave constraint is added to maintain the multimodal nature. The original five global optima are now infeasible and a new set of ten global optima become the solutions (where the constraint is active in all cases). The objective function to be minimised is:

$$f(x) = -\sin^6(5\pi x) + 1$$

subject to:

$$g_1(x) = -\cos(10\pi x) \leq 0$$

where the bounds are $0 \leq x \leq 1$. A plot of the function and constraint is given in figure 3

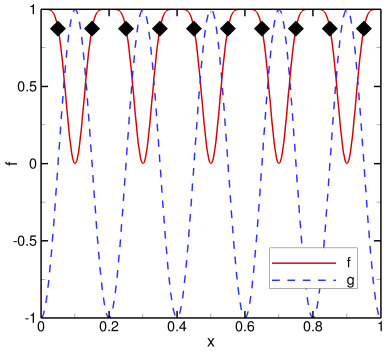


Figure 3: `F10` where black diamond symbols show location of optima

4.3. `F11` and `F12`

Functions `F11` and `F12` are constructed based on the Himmelblau function. Four constraints are added to the Himmelblau function for `F11`, while another four (to make eight in total) are added for `F12`. The four constraints added for `F11` (which are also used for `F12`) are quadratic functions where the active constraint lines are circles that pass through two of the Himmelblau function optima such that at any of the optimal solutions, two constraints are active, resulting in no change to the optimal solution. The four extra constraints present for `F12` are planar and pass through the four optimal solutions, forcing the optimal solutions to lie at the boundary of three active constraints in a minutely small feasible region, represented a particularly challenging problem for the multimodal algorithms.

The objective function to be minimised for both functions is:

$$f(\mathbf{x}) = (x_1^2 + x_2 - 11)^2 + (x_1 + x_2^2 - 7)^2 + 1$$

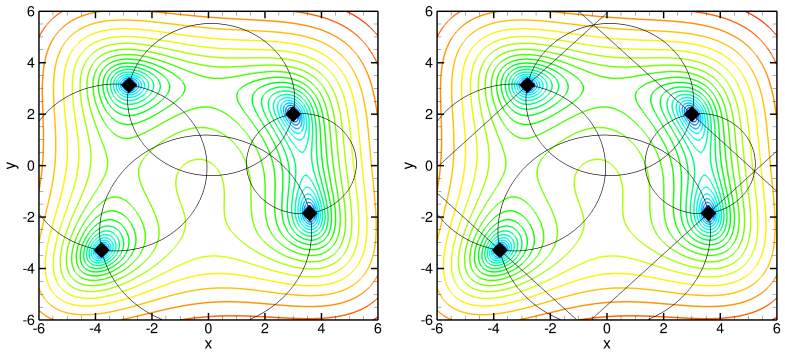
subject to:

$$\begin{aligned}
g_1(\mathbf{x}) &= \zeta_1 x_1 + \zeta_2 x_1 - \zeta_1\zeta_2 + \eta_1 x_2 + \eta_2 x_2 - \eta_1\eta_2 - x_1^2 - x_2^2 \leq 0 \\
g_2(\mathbf{x}) &= \zeta_2 x_1 + \zeta_3 x_1 - \zeta_2\zeta_3 + \eta_2 x_2 + \eta_3 x_2 - \eta_2\eta_3 - x_1^2 - x_2^2 \leq 0 \\
g_3(\mathbf{x}) &= \zeta_3 x_1 + \zeta_4 x_1 - \zeta_3\zeta_4 + \eta_3 x_2 + \eta_4 x_2 - \eta_3\eta_4 - x_1^2 - x_2^2 \leq 0 \\
g_4(\mathbf{x}) &= \zeta_4 x_1 + \zeta_1 x_1 - \zeta_4\zeta_1 + \eta_4 x_2 + \eta_1 x_2 - \eta_4\eta_1 - x_1^2 - x_2^2 \leq 0
\end{aligned}$$

where the further four constraints added for `F12` only are:

$$\begin{aligned}
g_5(\mathbf{x}) &= (x_1 - \zeta_1) + (x_2 - \eta_1) \leq 0 \\
g_6(\mathbf{x}) &= -(x_1 - \zeta_2) + (x_2 - \eta_2) \leq 0 \\
g_7(\mathbf{x}) &= -(x_1 - \zeta_3) - (x_2 - \eta_3) \leq 0 \\
g_8(\mathbf{x}) &= (x_1 - \zeta_4) - (x_2 - \eta_4) \leq 0
\end{aligned}$$

where $\boldsymbol{\zeta} = [3.0, -2.805, -3.779, 3.584]$ and $\boldsymbol{\eta} = [2.0, 3.131, -3.283, -1.848]$. The design space bounds are $-6 \leq x_d \leq 6$ $(d \in \{1, 2\})$. For function `F12`, two local optima also exist at the boundaries of where any two linear and one quadratic constraints intersect. Plots of the objective function with the active constraint lines are given in figure 4.



(a) `F11` (b) `F12`

Figure 4: Constrained Himmelblau functions where black diamond symbols show location of optima

4.4. `F13` to `F18`

For functions `F13` to `F18` a modified version of the Rastrigin function has been developed and is used. A modified version of the Rastrigin function has been presented by Deb and Saha [26], where the number of local optima in each dimension could be determined. Saha and Deb [69] also proposed a further modified version of the Rastrigin function where there are no local minima. In this paper, a combination of these two functions is proposed for the objective function of functions `F13` to `F18`, and a suitable constraint is also proposed. The resulting test problem has a specifiable number of local and global optima.

The objective function to be minimised is:

$$f(\mathbf{x}) = \sum_{d=1}^{D} \left\{ 10(1 + \cos(2\pi k_d x_d)) + 2k_d (x_d - 1)^2 H[x_d - 1] \right\}$$

subject to:

$$g(\mathbf{x}) = \sum_{d=1}^{D} 20 \cos(4\pi k_d x_d) \leq 0$$

where $H[]$ is the Heaviside function, which is given by:

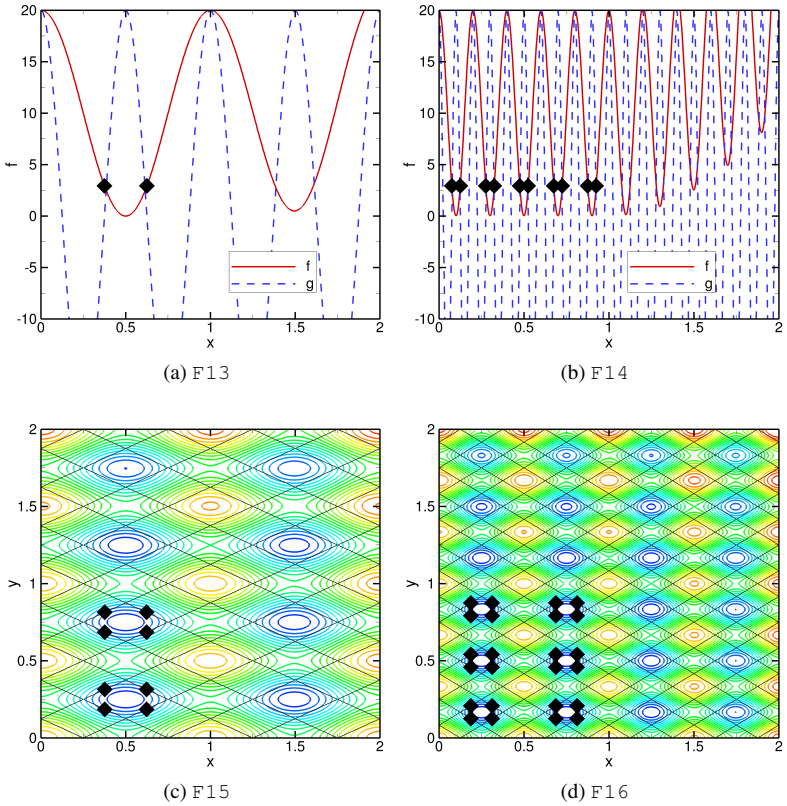
$$H[y] = \begin{cases} 1 & \text{if } y > 0 \\ 0 & \text{else} \end{cases}$$

The bounds of the problem are $0 \leq x_d \leq 2$ $(d \in \{1, \ldots, D\})$. k_d controls the number of global optima in each direction which are $2k_d$. A problem therefore has $2^D \prod_{d=1}^{D} k_d$ global optima, each having an objective function of $f = 10D - 5D\sqrt{2}$. The values of k_d for each problem are given in table 3. Figure 5 gives plots of the uni-dimensional (`F13` and `F14`) and bi-dimensional (`F15` and `F16`) problems.

The objective function is characterised by the region of the design space $0 \leq x_d \leq 1$, where all of the global optima are, and the region $1 \leq x_d \leq 2$ that contains local minima. The Heaviside function specifies that when $x_d \leq 1$, the function used is the global version of the Rastrigin function, whereas for $x_d > 1$, an underlying quadratic is added to the problem, leading to local minima.

Table 3: k_i values for functions

	F13	F14	F15	F16	F17	F18
k_1	1	5	1	2	1	1
k_2	-	-	2	3	1	1
k_3	-	-	-	-	2	1
k_4	-	-	-	-	-	1
k_5	-	-	-	-	-	2



(a) F13 (b) F14

(c) F15 (d) F16

Figure 5: Constrained modified-modulo-Rastrigin functions where black diamond symbols show location of optima

4.5. Performance Metrics

To analyse the performance of algorithms on the functions, a number of metrics are required that can be used to determine success, accuracy, feasibility and diversity. Mwaura *at al.* [70] outlined and analysed the common performance metrics used when analysing niching algorithms. Typically, a substantial number of independent runs of an algorithm on each function are performed, and overall metrics are determined based on the performance of each run.

The first metric used here is often the benchmark one to determine the success of a niching algorithm and is the peak ratio (PR). The PR of a result is given as the fraction of the number of global optima successfully identified against the number of global optima that exist, which for an individual run is:

$$PR_i = \frac{NGF_i}{N_g} \tag{18}$$

where NGF_i is the number of global optima successfully located to within a given tolerance level, ϵ, for run i. To determine NGF_i, the method given by Li *et al.* [68] is used but with an extra check to determine that a solution is feasible. This peak ratio calculation compares the fitness of solutions against the global optima fitness and as long as this is within a specified tolerance, ϵ, and no other solutions are within the vicinity of the current solution, then a new solution is assume to have been found. The peak ratio can be calculated throughout the run and can also be calculated at the end, after all runs have been performed. The overall PR over a number of independent runs is simply the mean average of the PR of each run:

$$PR = \frac{1}{NR} \sum_{i=1}^{NR} PR_i \tag{19}$$

The success rate, SR, which gives an overall measure of the success of each run is also used. This is calculated as:

$$SR = \frac{NSR}{NR} \tag{20}$$

where NSR is the number of successful runs. A successful run is characterised as one where all of the global optima have been located, which is a run that has $PR_i = 1$.

The peak ratio (and success rate) give a measure of understanding the success of a given algorithm in terms of a ability to locate optima. However, particularly when dealing with niching, it is also useful to consider the diversity within the population. To do this, two metrics can be compared which give a global measure of the convergence to optima, as well as the convergence of niches. To determine the convergence to the optima the peak accuracy, PA, which is related to the peak ratio, is used. The peak accuracy gives a way of quantifying the average error of the objective of the optima found by the niching algorithm, and for each run is given as:

$$PA_i = \frac{1}{N_g} \sum_{1}^{N_g} \min_{n \in \{1,\ldots,N\}} |f(\mathbf{x}^*) - f(\mathbf{x}_n)| \tag{21}$$

The overall peak accuracy is then the mean average over the NR runs. To determine the convergence of niches it is useful to consider the average feasible objective error, AOV:

$$AOV_i = \frac{1}{N_f} \sum_{n=1}^{N_f} |f(\mathbf{x}^*) - f(\mathbf{x}_n)| \tag{22}$$

where N_f is the number of feasible individuals at the end of a run which is then averaged over the NR runs. In a fully converged population, all of the individuals will converge to multiple niches, where one niche exists for each of the global optima.

All well as considering the convergence and diversity within the function space, it is also important to consider the same within the design space. To determine the convergence onto the optima within the design space, the peak distance, PD, measures the average distance of each peak to the nearest individual:

$$PD_i = \frac{1}{N_g} \sum_{j=1}^{N_g} \min_{n \in \{1,\ldots,N\}} \|\mathbf{x}^{*j} - \mathbf{x}_n\|_2 \tag{23}$$

which is then averaged over the NR runs. The diversity of the population is determined by evaluating the average distance to the nearest neighbour ($ADNN$):

$$ADNN_i = \frac{1}{N} \sum_{n=1}^{N} \min_{k \in \{1,\ldots,N\}, k \neq n} \|\mathbf{x}_k - \mathbf{x}_n\|_2 \tag{24}$$

which is again averaged over the NR runs.

To give an indication of the speed of the algorithms, the convergence speed is also used. The convergence speed is determined as the mean average number of function evaluations required to locate all global optima to within a

tolerance, ϵ. The convergence speed is checked at each iteration to determine whether all of the global optima have been located, hence it is possible for an algorithm, if it cannot maintain stable niches, to then lose some of these optima. Despite this, the convergence speed will still return a number of function evaluations that is less than the maximum. If the run finished without locating all of the global optima, then when calculating the average, the maximum number of functions evaluations is used, as per Li *et al.* [68]. The overall convergence speed (OCS) is the average of the convergence speed over all functions, hence gives a global measure of overall convergence for a specific algorithm.

Finally, to quantify the performance in terms of constraints, the feasible fraction (FF) is calculated. The FF gives the fraction of the number of feasible individuals at the end of a run to the total number of individuals:

$$FF_i = \frac{N_f}{N} \tag{25}$$

The overall FF over a number of runs is obtained by averaging the FF from each run.

5. Canonical DE-Based Constrained Niching Algorithms

To investigate multimodal optimization of constrained functions, a number of conventional niching algorithms that use DE and have been shown to perform well at unconstrained multimodal optimization, are combined with a common constraint handling method also often applied to DE to produce new, constrained niching techniques. In this section, these are fully outlined.

The DE-based niching algorithms that are developed into constrained niching algorithms are: canonical DE, DE using nrand1 mutation (NRAND1) [59], DE using inrand1/r (nearest neighbour with ring network) mutation (INRAND1) [60], crowding DE (CDE) [18], neighbourhood-based CDE (NCDE) [22], species-based DE (SDE) [56], neighbourhood-based SDE (NSDE) [22] and fitness-sharing DE (SHDE) [18]. Initially, DE, CDE, NCDE, SDE, NSDE, SHDE all use the canonical form of the DE algorithm with binomial crossover and the rand/1 mutation strategy. Later in this work, the best/1 mutation strategy is also considered as well as more advanced DE-based algorithms (SHADE and L-SHADE).

Constraints are handled using the feasibility rules of Deb [14], which state that when choosing between two locations, if both locations are feasible then the one with best fitness wins, otherwise a feasible location is preferred, otherwise the one with the smallest constraint violation wins. This can be written as a domination operator, where, given two locations $\mathbf{x}_a$ and $\mathbf{x}_b$, $\mathbf{x}_b$ dominates $\mathbf{x}_a$ based on the following:

$$\mathbf{x}_a \prec \mathbf{x}_b \Leftrightarrow \begin{cases} f(\mathbf{x}_b) < f(\mathbf{x}_a) & \text{and} \quad \phi(\mathbf{x}_a), \phi(\mathbf{x}_b) = 0 \\ \phi(\mathbf{x}_b) = 0 & \text{and} \quad \phi(\mathbf{x}_a) > 0 \\ \phi(\mathbf{x}_b) < \phi(\mathbf{x}_a) & \text{and} \quad \phi(\mathbf{x}_a), \phi(\mathbf{x}_b) > 0 \end{cases} \tag{26}$$

where ϕ is the constraint violation given by:

$$\phi(\mathbf{x}) = \sum_{i=1}^{p} \max[0, g_i(\mathbf{x})] + \sum_{j=1}^{q} |h_j(\mathbf{x})|$$

In DE, these feasibility rules are commonly used in the selection step to determine whether the trial vector should replace the target vector. Hence, rewriting equation 7 for constrained optimization leads to:

$$\mathbf{x}_n(t+1) = \begin{cases} \mathbf{u}_n & \text{if } \mathbf{x}_n(t) \prec \mathbf{u}_n \\ \mathbf{x}_n(t) & \text{otherwise} \end{cases} \tag{27}$$

The new constrained niching techniques developed for this work are all termed:

- Feasible DE (fDE)[5], which is based on canonical DE
- Feasible DE using nrand1 mutation (fNRAND1),

[5]First presented by Mezura-Montes *et al.* [71]

- Feasible DE using inrand1/r (nearest neighbour with ring network) mutation (fINRAND1),
- Feasible crowding DE (fCDE), which is based on the CDE algorithm [18]
- Feasible neighbourhood-based CDE (fNCDE), which is based on the NCDE algorithm [22]
- Feasible species-based DE (fSDE), which is based on the SDE algorithm [56]
- Feasible neighbourhood-based SDE (fNSDE), which is based on the NSDE algorithm [22]
- Feasible fitness-sharing DE (fSHDE) , which is based on the SHDE algorithm [18]

Each is outlined below.

5.1. fDE

fDE uses the canonical DE algorithm with equation 27 used for the selection stage. The rand/1/bin strategy is used. The overall algorithm is outlined in algorithm 4.

Algorithm 4 fDE algorithm

```
Randomly initialise individuals and calculate objective
while FEs<FEs_max do
    for n = 1 → N do
        Perform rand/1 mutation: equation 4
        Perform binomial crossover: equation 6
        Calculate objective and constraints of trial vector
    end for
    for n = 1 → N do
        Update n-th target vector: equation 27
    end for
end while
```

5.2. fNRAND1

The fNRAND1 algorithm uses the target vector of the n-th individual's nearest neighbour (in the design space), $\mathbf{x}_{NN_n}$, as the base vector against which to provide the difference vector in the mutation stage. The mutation is given as:

$$\mathbf{v}_n = \mathbf{x}_{NN_n}(t) + F(\mathbf{x}_{r_1}(t) - \mathbf{x}_{r_2}(t)) \tag{28}$$

where:

$$NN_n = \underset{i \in \{1,\ldots,N\}, i \neq n}{\arg\min} \|\mathbf{x}_n - \mathbf{x}_i\|_2$$

The selection stage uses equation 27 for feasibility. The overall algorithm is outlined in algorithm 5.

5.3. fINRAND1

The fINRAND1 algorithm uses the target vector of n-th individual's nearest neighbour (in the design space) within its local neighbourhood, $\mathbf{x}_{INN_n}$, as the base vector against which to provide the difference vector in the mutation stage. In this work, an index-based ring neighbourhood is used hence this reduces computational complexity against fNRAND1. The mutation is given as:

$$\mathbf{v}_n = \mathbf{x}_{INN_n}(t) + F(\mathbf{x}_{r_1}(t) - \mathbf{x}_{r_2}(t)) \tag{29}$$

where:

Algorithm 5 fNRAND1 algorithm

```
Randomly initialise individuals and calculate objective
while FEs<FEs_max do
    for n = 1 → N do
        Find the nearest neighbour to x_n
        Perform nrand/1 mutation: equation 28
        Perform binomial crossover: equation 6
        Calculate objective and constraints of trial vector
    end for
    for n = 1 → N do
        Update n-th target vector: equation 27
    end for
end while
```

$$INN_n = \begin{cases} \underset{i \in \{N,2\}}{\arg\min} \|\mathbf{x}_n - \mathbf{x}_i\|_2 & \text{if } n = 1 \\ \underset{i \in \{N-1,1\}}{\arg\min} \|\mathbf{x}_n - \mathbf{x}_i\|_2 & \text{if } n = N \\ \underset{i \in \{n-1,n+1\}}{\arg\min} \|\mathbf{x}_n - \mathbf{x}_i\|_2 & \text{otherwise} \end{cases}$$

The selection stage uses equation 27 for feasibility. The overall algorithm is outlined in algorithm 6.

Algorithm 6 fINRAND1 algorithm

```
Randomly initialise individuals and calculate objective
while FEs<FEs_max do
    for n = 1 → N do
        Find the nearest neighbour to x_n in ring neighbourhood
        Perform inrand/1 mutation: equation 29
        Perform binomial crossover: equation 6
        Calculate objective and constraints of trial vector
    end for
    for n = 1 → N do
        Update n-th target vector: equation 27
    end for
end while
```

5.4. fCDE

The fCDE algorithm uses the normal CDE algorithm but with the feasible selection method. CDE requires creating a trial vector by the chosen mutation strategy, and once this is found, the closest individual to the trial vector (in the design space), $\mathbf{x}_{u_n}$, needs to be found. Once this is found, this closest individual is replaced by the trial vector if the trial vector is better, determined using the feasibility rules (this is where fCDE differs from CDE):

$$\mathbf{x}_{u_n}(t+1) = \begin{cases} \mathbf{u}_n & \text{if } \mathbf{x}_{u_n}(t) \prec \mathbf{u}_n \\ \mathbf{x}_{u_n}(t) & \text{otherwise} \end{cases} \tag{30}$$

The fCDE algorithm is outlined in algorithm 7.

5.5. fNCDE

The feasible neighbourhood algorithm of crowding DE, fNCDE, generates a trial vector from a neighbourhood that is made up from the m nearest individuals to the n-th individual, in the design space. Hence, when performing rand/1 mutation (equation 4), r_1, r_2 and r_3 are uniformly distributed random integers that come from the set of integers that

Algorithm 7 fCDE algorithm

```
Randomly initialise individuals and calculate objective
while FEs<FEs_max do
    for n = 1 → N do
        Perform rand/1 mutation: equation 4
        Perform binomial crossover: equation 6
        Calculate objective and constraints of trial vector
    end for
    for n = 1 → N do
        Find the closest individual to u_n
        Update closest individual: equation 30
    end for
end while
```

represent the m nearest neighbours. Once a trial vector is found, updating occurs on the whole population according to normal crowding DE, where the nearest neighbour to the trial vector is used for comparison. The algorithm is given in algorithm 8.

Algorithm 8 fNCDE algorithm

```
Randomly initialise individuals and calculate objective
while FEs<FEs_max do
    for n = 1 → N do
        Find the nearest m individuals to x_n
        Perform rand/1 mutation using the nearest m individuals
        Perform binomial crossover: equation 6
        Calculate objective and constraints of trial vector
    end for
    for n = 1 → N do
        Find the closest individual to u_n in entire population
        Update closest individual: equation 30
    end for
end while
```

5.6. fSDE

The SDE algorithm creates neighbourhoods (or species) based on the closeness of fit individuals. First, individuals must be sorted, which in SDE is done in ascending order based on fitness. However, in fSDE this is performed using the feasibility rules so feasible individuals are sorted by their fitness in ascending order, who all precede infeasible individuals, who are sorted by their constraint violation in ascending order. This results in a sorted list of individuals where the first individual in the list will be the fittest (if at least one individual is feasible), while the last will be the one who most violates the constraints (assuming at least one individual is infeasible). If all individuals are feasible or all are infeasible, then the list is a sorted list of fitness or constraint violation, respectively.

Once a sorted list has been found, species are determined based on the distance (in the design space) from a species seed, where the seed radius σ determines the spread that species can have. If a species has less than m (a user-determined constant) individuals then extra individuals are added to the population to ensure that all species have equal numbers of individuals. When performing rand/1 mutation, r_1, r_2 and r_3 are uniformly distributed random integers that come from the set of integers that represent the m individuals of the species that the n-th individual belongs to. It should be noted that individuals can belong to multiple species.

Finally, since the population has been increased, only the N fittest individuals are kept for the next iteration, determined again by feasibility rules. The overall algorithm is outlined in algorithm 9.

Algorithm 9 fSDE algorithm

Randomly initialise individuals and calculate objective
while FEs<FEs$_{max}$ **do**
  Generate species: algorithm 10
  for $n = 1 \rightarrow N$ **do**
    Perform rand/1 mutation using individuals within the species of n-th individual
    Perform binomial crossover: equation 6
    Calculate objective and constraints of trial vector
    If donor fitness is same as its species seed, then randomly generate new trial vector
  end for
  for $n = 1 \rightarrow N$ **do**
    Update n-th target vector: equation 27
  end for
  Compare individuals using feasibility rules and keep N fittest individuals
end while

Algorithm 10 fSDE species generation algorithm

Sort individuals based on feasibility rules
Sorted individuals are assigned to possible candidate solutions
First species seed is best candidate solution - remove that solution from candidates
for $n = 1 \rightarrow N$ **do**
  for $s = 1 \rightarrow$ number of species **do**
    if n-th candidate entry is not empty and is less that r_s away from s-th seed **then**
      Solution is not a new seed
      Note solution is in s-th species
    end if
  end for
  if n-th candidate entry is new seed **then**
    Increment number of species
    Store n-th candidate solution as seed and remove from list of candidates
  end if
end for
for $s = 1 \rightarrow$ number of species **do**
  If the s-th species has less that m individuals, randomly generate new individuals within radius of species seed
end for

5.7. fNSDE

The fNSDE algorithm is similar, though slightly less intricate compared to fSDE. The primary difference is that when generating species, the species seed is the most fit individual that does not currently belong to another species, and all individuals within that species are the closest m who also do not yet belong to another species. There is therefore always m individuals in all species. The overall algorithm is given in algorithm 11.

5.8. fSHDE

The final algorithm developed and used here is fSHDE. This involves using fitness sharing which divides the population based on similarity and penalises individuals who are close to each other. The algorithm progresses mostly as the canonical DE, except that once the trial is generated, instead of performing selection, it is added to the population. The population therefore grows by a factor of two. On this super-population, the fitness of individuals is shared, such that the shared fitness becomes:

$$f'(\mathbf{x}_n) = \frac{f(\mathbf{x}_n)}{\sum_{i=1}^{2N} \lambda_{i,n}} \tag{31}$$

where

Algorithm 11 fNSDE algorithm

Randomly initialise individuals and calculate objective
while FEs<FEs$_{max}$ **do**
  Generate species: algorithm 12
  for $n = 1 \rightarrow N$ **do**
    Perform rand/1 mutation using individuals within the species of n-th individual
    Perform binomial crossover: equation 6
    Calculate objective and constraints of trial vector
    If trial fitness is same as its species seed, then randomly generate new trial vector
  end for
  for $n = 1 \rightarrow N$ **do**
    Update n-th target vector: equation 27
  end for
end while

Algorithm 12 fNSDE species generation algorithm

Sort individuals based on feasibility rules
Sorted individuals are assigned to possible candidate solutions
for $s = 1 \rightarrow \text{floor}(N/m)$ **do**
  First element of s-th species is best individual in candidate solutions
  for $i = 2 \rightarrow m$ **do**
    Determine closest individual in candidate solutions to first element of s-th species
    Add that individual to the s-th species and remove from list of candidates
  end for
end for
if $N \text{mod } m \neq 0$ **then**
  Final species contains remaining candidate solutions
end if

$$\lambda_{i,n} = \begin{cases} 1 - \left(\frac{\|\mathbf{x}_i - \mathbf{x}_n\|_2}{\sigma}\right)^{\alpha}, & \text{if} \|\mathbf{x}_i - \mathbf{x}_n\|_2 < \sigma \\ 0, & \text{otherwise} \end{cases} \tag{32}$$

where σ is the sharing radius, and α is the sharing level (set to 1.0 in this paper[18]).

Once the fitness has been shared, the N fittest individuals are kept. This is determined based on a feasibility rules selection where the fitness used is the shared fitness. To ensure the previous best solution is kept, if it was removed during the selection process then it replaces the new worst solution. The overall algorithm is shown in algorithm 13.

Algorithm 13 fSHDE algorithm

Randomly initialise individuals and calculate objective
while FEs<FEs$_{max}$ **do**
  for $n = 1 \rightarrow N$ **do**
    Perform rand/1 mutation: equation 4
    Perform binomial crossover: equation 6
    Calculate objective and constraints of trial vector
    Add trial vector to population: $\mathbf{x}_{N+n} = \mathbf{u}_n$
  end for
  Calculate shared fitness of enlarged population
  Keep N (shared) fittest individuals
  If previously best solution is lost, replace worst individual with previous best
end while

5.9. Time Complexity

Before progressing to consider the performance of the algorithms, the time complexity of each is discussed. Since the constrained niching algorithms differ from their unconstrained forms by primarily the use of the feasibility rules, the time complexity of each is the same as the unconstrained form assuming $q \ll N$ (where q is the number of constraints).

The complexity of the eight algorithms is given in table 4. Hence, the only two algorithms able to maintain the complexity of the underlying DE algorithm are fDE (this is DE with the feasibility rules added) and fINRAND1. All others are $O(N^2)$ at worst. In the cases of fNRAND1, fCDE, fNCDE and fNSDE, the $O(N^2)$ complexity comes about due to having to perform nearest nighbour searches for each individual in the population, while for fSHDE, it is due to having to calculate the shared fitness for each individual. The fSDE algorithm is the exception to high complexity. If the number of species is much less that the number of individuals in the whole population (this would be the case if a number of tightly packed niches were formed), then the complexity is $O(N)$. On the other hand, if the number of species were the same as the number of individuals (for example, if the population was sparsely spread) then fSDE increases to $O(N^2)$ complexity.

Table 4: Complexity of constrained niching algorithms

Algorithm	Complexity
fDE	$O(N)$
fNRAND1	$O(N^2)$
fINRAND1	$O(N)$
fCDE	$O(N^2)$
fNCDE	$O(N^2)$
fSDE	$O(N)$ to $O(N^2)$
fNSDE	$O(N^2)$
fSHDE	$O(N^2)$

5.10. Parameter Tuning

The performance of DE-based algorithms is highly dependent on the parameter values chosen [67]. As such, before presenting results for the algorithms, a tuning process of the parameters for the algorithms individually needs to occur.

For this tuning process, the eight constrained niching DE algorithms have been run on the low-dimensional analytical function suite. The number of runs on each function, NR, is 50 for each of the eight algorithms. The population of each algorithm is set dependent on the specific problem, and is given by $N = 40\sqrt{DN_g}$ [72]. For the i-th run of each algorithm, the population is initialised from the same location to ensure that the performance is determined by the internal search mechanism, rather than the initial location of the population, where the initial location for each run is randomly set between the design space bounds. If at any point an individual goes beyond the bounds, it is randomly reinitialised back in the design space. For this parameter tuning, FEs_{max} is 400,000. The effect of a smaller number of maximum function calls is investigated later.

For each of the eight algorithms, the values of F and CR are tuned with a high and low value for each considered. For fSDE and fSHDE the sharing radius, σ, is also tuned (this is given as a percentage of the size of the design space) while for fSDE, fNCDE and fNSDE the species/neighbourhood size, m, is tuned. The values of the parameters considered in the tuning process are given in table 6

The process used to choose the tuned parameters is as follows. First, each algorithm is run 50 times on the suite of low-dimensional benchmark functions for combinations of the parameter settings. Then, for each algorithm, for each combination of parameter settings, a Wilcoxon rank-sum test[73] is performed on the peak ratio results. In this situation, the rank-sum test is used to test the null hypothesis that "*the median of the peak ratios of algorithms with different settings are equal*". The confidence level of the test is 95%. Both right-tailed and left-tailed p-values are calculated to test the alternate hypotheses of whether the median of A is greater that the median of B and whether the median of A is less that the median of B, respectively. If the median of an algorithm with a specific setting is greater

Table 5: Parameter values used for tuning of canonical DE-based constrained niching algorithms

		Algorithm							
Param.	Values	fCDE	fDE	fINRAND1	fNCDE	fNRAND1	fNSDE	fSDE	fSHDE
CR	0.1, 0.9	✓	✓	✓	✓	✓	✓	✓	✓
F	0.1, 0.9	✓	✓	✓	✓	✓	✓	✓	✓
σ	0.1%, 1%, 10%							✓	✓
m	5, 10, 20				✓		✓	✓	

than another setting, then the first setting (parameter values) is said to have won. For each specific setting, the number of other settings that it wins and loses against are totalled and used as a basis to create a rank. For a specific algorithm, a rank of 1 is the setting with the highest number of wins, and so on. If multiple settings have the same number of wins, then the one with the lowest OCS[6] is best. For example, table S1 shows the results of the tuning process for fCDE, where four different combinations of F and CR have been run. Three of the settings have one win (all beating the remaining setting, but none beat each other), so the one with best rank is the one with lowest OCS (in this case, $F = 0.1$, $CR = 0.1$). The results of this process are given in tables S1 to S13. Table 6 gives a summary of the final tuned parameters for each of the algorithms.

Table 6: Tuned parameter values used for canonical DE-based constrained niching algorithms

	Algorithm							
Param.	fCDE	fDE	fINRAND1	fNCDE	fNRAND1	fNSDE	fSDE	fSHDE
CR	0.1	0.1	0.9	0.9	0.9	0.1	0.9	0.1
F	0.1	0.9	0.9	0.9	0.9	0.9	0.9	0.1
σ							10%	0.1%
m				10		10	20	

6. Results on Low-Dimensional Suite

The eight canonical DE-based constrained niching algorithms have been run on the low-dimensional analytical function suite and the results are presented here.

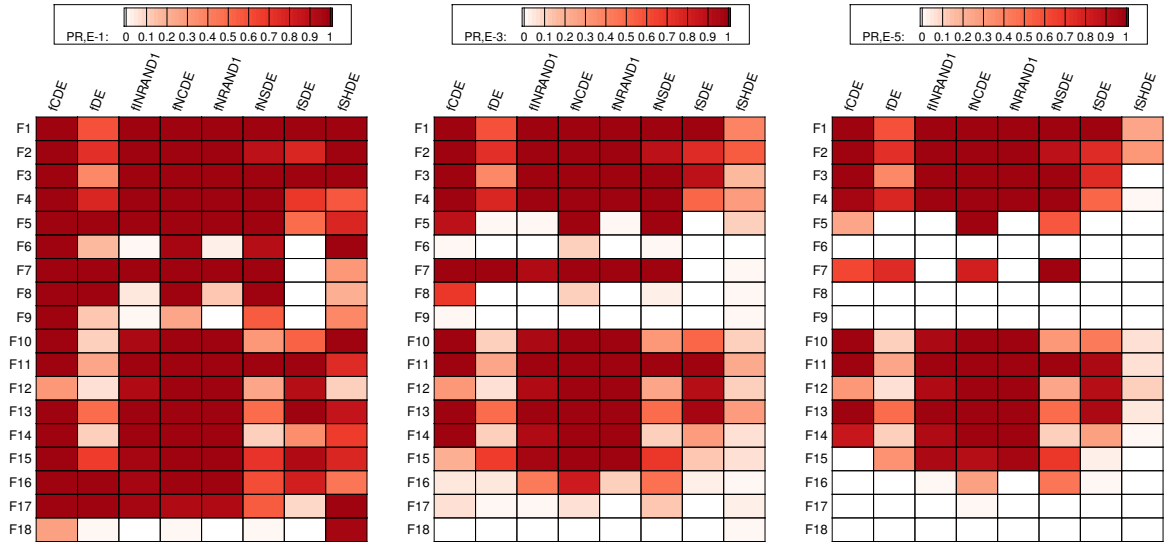
The parameters and algorithm settings are all the same as during the parameter tuning process (see section 5.10 for details). However, an investigation by Piotrowski *et al.* [74] into the effect of having a pre-determined number of function calls on the relative results of different global algorithms revealed that the relative performance of algorithms with maximum allowable function calls. As such, two investigations are presented here: the first uses a high number of pre-determined function calls ($\text{FEs}_{max} = 400,000$), while the second uses a restricted number of function calls which for each test function is given by $\text{FEs}_{max} = 2000D\sqrt{N_g}$, as suggested by Qu *et al.* [75]. For the low-dimensional function suite, the restricted values of FEs_{max} range from 2,828 to 80,000.

6.1. High FEs_{max}

The peak ratio and success rates for all the algorithms tested on each function with a high allowable number of FEs_{max} are presented in tables S14 and S15. From this raw data, figure 6 provides an overview of the peak ratios at $\epsilon = 1 \times 10^{-1}, 1 \times 10^{-3}, 1 \times 10^{-5}$. Overall, it is clear that the majority of results on functions `F1` to `F4` and `F10` to `F14` are positive (mostly have a high PR). These functions are all lower dimensional problems with few global optima, so this is expected. Functions `F8`, `F9`, `F17` and `F18` all have, almost universally with all algorithms, generally less

[6] See section 4.5 for the definition of OCS, where the tolerance used is $\epsilon \leq 1E-4$.

good performance with no algorithm able to locate any optima (PR of 0) to within a tolerance of 1×10^{-5}. These functions all have higher numbers of dimensions and/or higher numbers of global optima so represent particularly difficult problems.



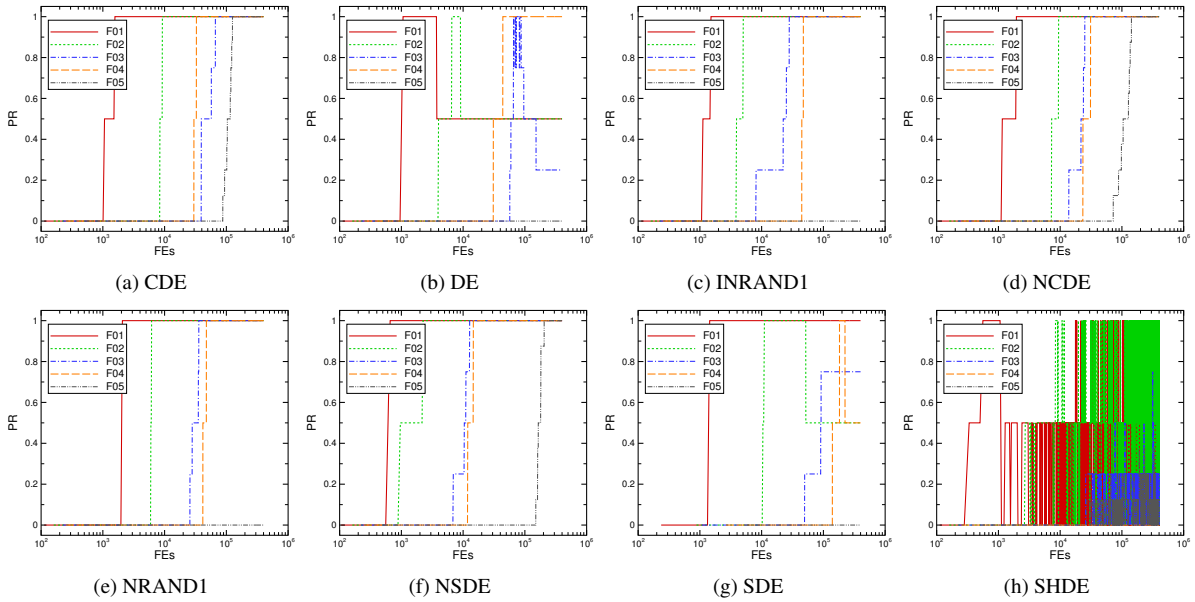
(a) $\epsilon = 1 \times 10^{-1}$ (b) $\epsilon = 1 \times 10^{-3}$ (c) $\epsilon = 1 \times 10^{-5}$

Figure 6: Overview of peak ratios for canonical DE-based algorithms run on low-dimensional suite with high FEs$_{max}$

The convergence speeds are given in table 7, while figures 7 to 10 show the convergence of the PR at a tolerance level of $\epsilon = 1 \times 10^{-3}$ for all of the algorithms. First, in terms of the convergence speeds, the two neighbourhood based algorithms (fNSDE and fNCDE) have the fastest overall convergence rates across the board. However, while fNSDE has little difficulty in finding all of the optima quickly, the niches that are formed appear unstable. This is particularly emphasised in the convergence plots, which show that for function `F11`, for example, fNSDE locates all of the optima rapidly and then can maintain these niches. However, for function `F14`, the majority of optima are located, but the niches are not able to be maintained and only one global optima results. On the other hand, fNCDE locates optima at a slightly slower rate than fNSDE, but is much better able to maintain stable niches.

Table 7: Convergence speeds (with $\epsilon = 1 \times 10^{-4}$) for canonical DE-based algorithms run on low-dimensional suite with with high FEs_{max} (standard deviations given in parentheses)

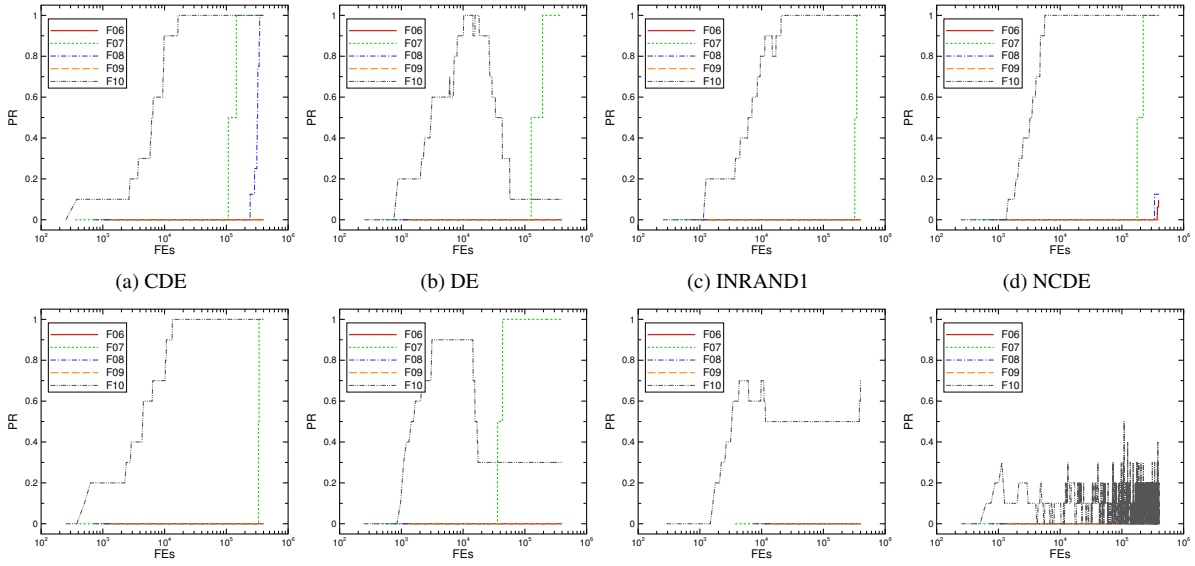
Func	fCDE	fDE	fINRAND1	fNCDE	fNRAND1	fNSDE	fSDE	fSHDE
F1	6677 (1693)	10016 (55712)	2552 (451)	4379 (1073)	3436 (708)	979 (110)	1839 (202)	71128 (56262)
F2	18862 (3519)	23921 (76776)	6940 (773)	11726 (2635)	8321 (651)	3350 (498)	34135 (74967)	49158 (40464)
F3	127457 (14290)	103052 (88059)	42117 (3823)	42892 (3531)	60039 (4898)	21981 (4866)	263556 (136775)	400000 (0)
F4	78237 (54771)	59775 (49349)	57952 (4457)	41689 (4039)	64103 (3821)	16066 (3759)	302328 (102173)	392763 (50655)
F5	309278 (104005)	400000 (0)	400000 (0)	171982 (25857)	400000 (0)	373532 (30433)	400000 (0)	400000 (0)
F6	400000 (0)	400000 (0)	400000 (0)	400000 (0)	400000 (0)	400000 (0)	400000 (0)	400000 (0)
F7	262017 (81681)	320083 (40209)	400000 (0)	291201 (27089)	400000 (0)	83748 (24277)	400000 (0)	400000 (0)
F8	400000 (0)	400000 (0)	400000 (0)	400000 (0)	400000 (0)	400000 (0)	400000 (0)	400000 (0)
F9	400000 (0)	400000 (0)	400000 (0)	400000 (0)	400000 (0)	400000 (0)	400000 (0)	400000 (0)
F10	54371 (7173)	400000 (0)	176788 (182441)	15337 (7891)	30731 (4270)	34866 (107673)	400000 (0)	400000 (0)
F11	120241 (14868)	392944 (49387)	24828 (2663)	23699 (2905)	31468 (2051)	5909 (812)	78871 (117529)	400000 (0)
F12	400000 (0)	400000 (0)	142406 (153781)	46812 (11631)	63720 (5946)	11598 (2407)	199649 (135749)	400000 (0)
F13	20316 (3391)	50561 (129039)	5996 (1492)	8645 (2364)	8542 (1136)	1541 (692)	56598 (109150)	341708 (93328)
F14	257627 (27428)	400000 (0)	256726 (168742)	59869 (60842)	74246 (6487)	400000 (0)	400000 (0)	400000 (0)
F15	400000 (0)	395168 (13896)	291644 (67597)	283289 (90678)	293596 (31181)	395129 (27420)	400000 (0)	400000 (0)
F16	400000 (0)	400000 (0)	400000 (0)	400000 (0)	400000 (0)	400000 (0)	400000 (0)	400000 (0)
F17	400000 (0)	400000 (0)	400000 (0)	400000 (0)	400000 (0)	400000 (0)	400000 (0)	400000 (0)
F18	400000 (0)	400000 (0)	400000 (0)	400000 (0)	400000 (0)	400000 (0)	400000 (0)	400000 (0)
Mean	247505	297529	233775	188973	213234	208261	296499	358598



(a) CDE (b) DE (c) INRAND1 (d) NCDE

(e) NRAND1 (f) NSDE (g) SDE (h) SHDE

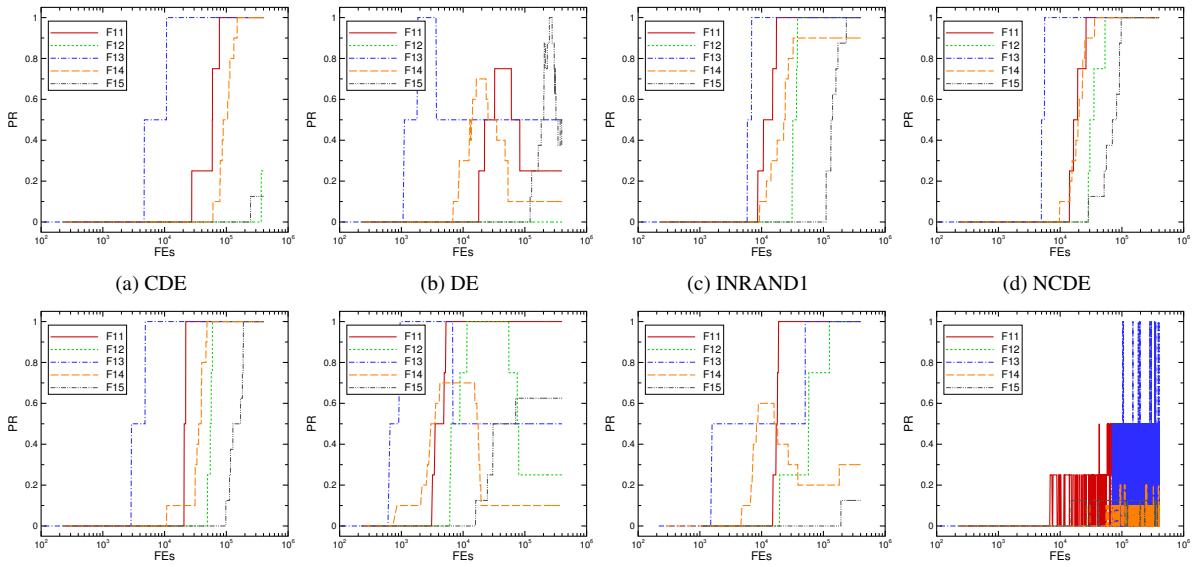
Figure 7: Convergence of peak ratios for $\epsilon = 1E-3$ of functions F01 to F05



(a) CDE (b) DE (c) INRAND1 (d) NCDE

(e) NRAND1 (f) NSDE (g) SDE (h) SHDE

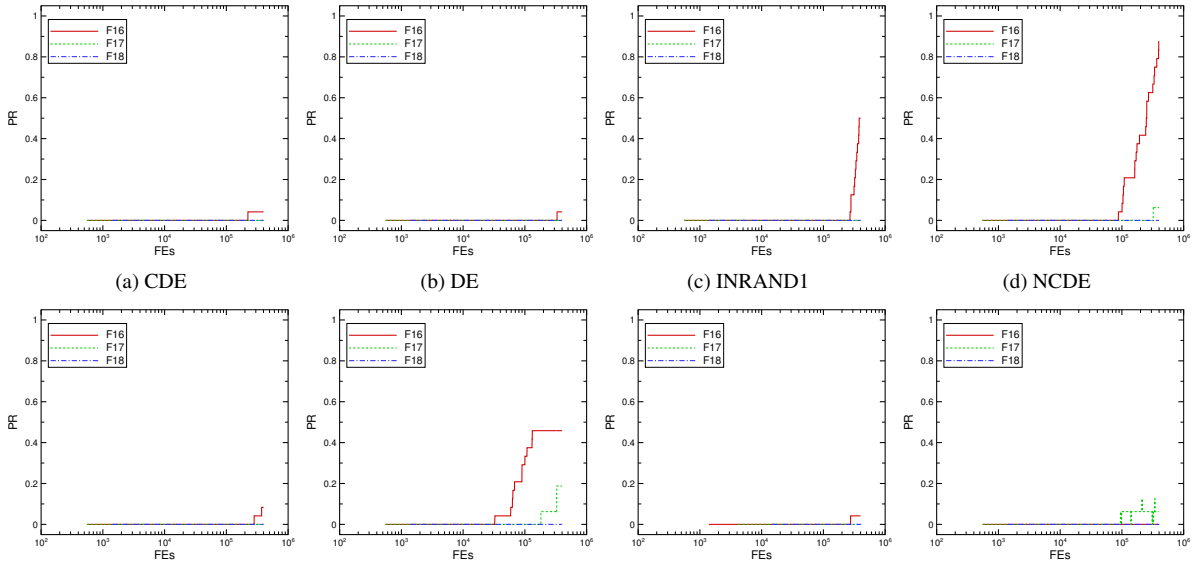
Figure 8: Convergence of peak ratios for $\epsilon = 1E-3$ of functions `F06` to `F10`



(a) CDE (b) DE (c) INRAND1 (d) NCDE

(e) NRAND1 (f) NSDE (g) SDE (h) SHDE

Figure 9: Convergence of peak ratios for $\epsilon = 1E-3$ of functions `F11` to `F15`

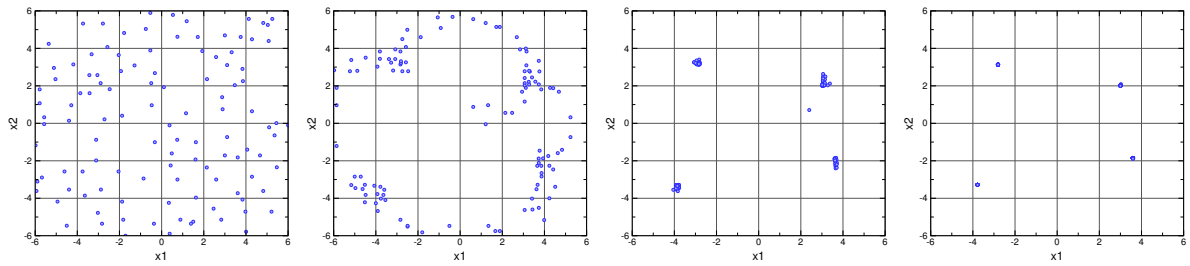


(a) CDE (b) DE (c) INRAND1 (d) NCDE

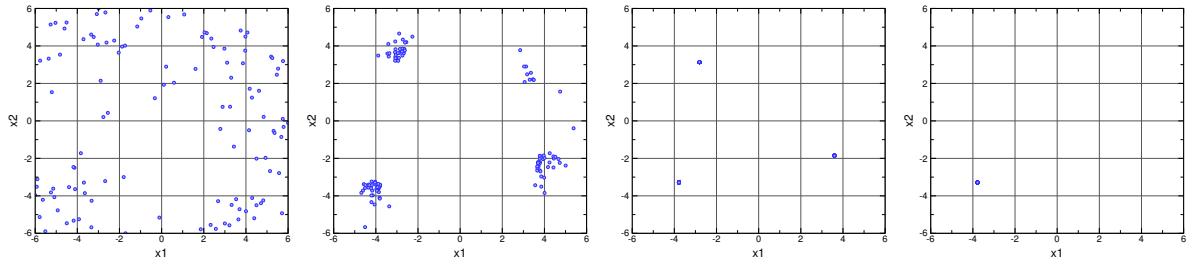
(e) NRAND1 (f) NSDE (g) SDE (h) SHDE

Figure 10: Convergence of peak ratios for $\epsilon = 1E-3$ of functions `F16` to `F18`

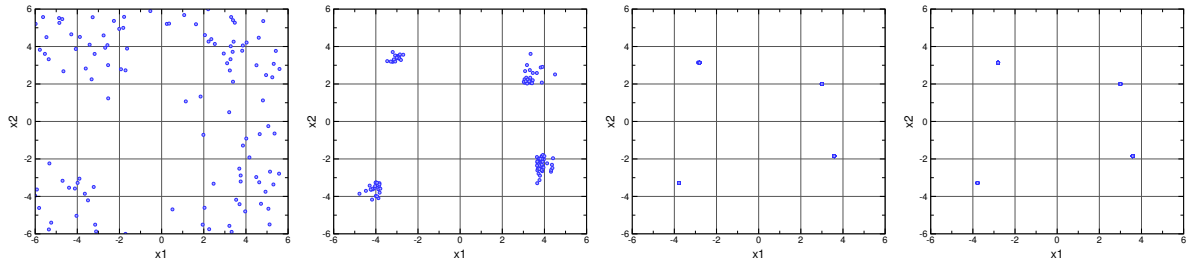
Using the regular DE (fDE) without any niching algorithm has yielded some reasonable results. DE typically has a tendency to cluster anyway, often without the need for explicit niching techniques [59], and this is, somewhat, observed here. The average convergence speed (table 7) of fDE is lower than fCDE, fINRAND1, fSDE and fSHDE, which requires (sometimes considerably) greater complexity compared to fDE. In terms of algorithmic development, fINRAND1 and fNRAND1 require the least effort to develop, with only a few extra lines of code added and no change in code logic, and of these, fNRAND1 appears to give better performance both in terms of convergence speeds, peak ratios and success rates than fDE, however, since a nearest neighbour search is required for fNRAND1, the algorithm complexity can be up to $O(N^2)$ per iteration, compared to $O(N)$ for fDE. However, fDE can have difficulty maintaining stable niches and appears to prefer converging to a single solution. For example, figures 11 and 12 shows the locations of individuals at snapshots during the optimization; fDE appears to be converging to four different location but then tends to cluster the entire population together.



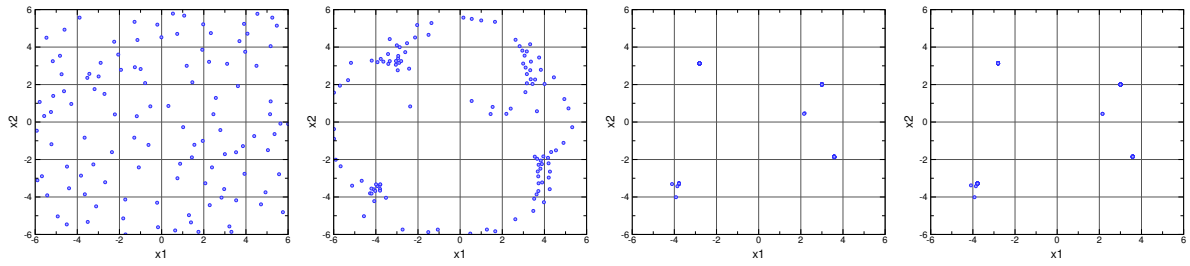
(a) fCDE: 400 FEs (b) fCDE: 4,000 FEs (c) fCDE: 40,000 FEs (d) fCDE: 400,000 FEs



(e) fDE: 400 FEs (f) fDE: 4,000 FEs (g) fDE: 40,000 FEs (h) fDE: 400,000 FEs

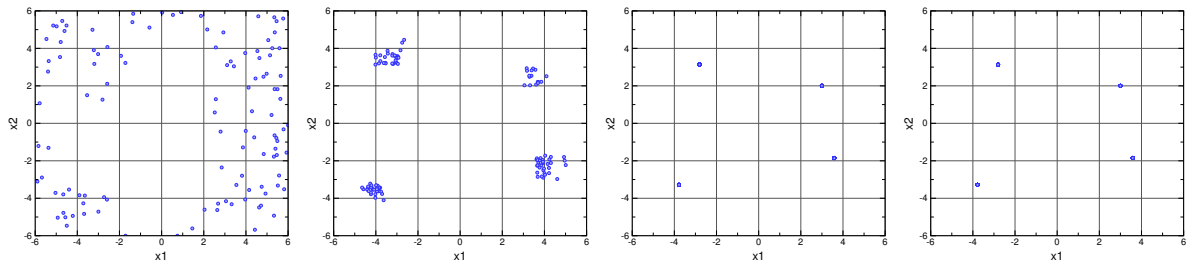


(i) fINRAND1: 400 FEs (j) fINRAND1: 4,000 FEs (k) fINRAND1: 40,000 FEs (l) fINRAND1: 400,000 FEs

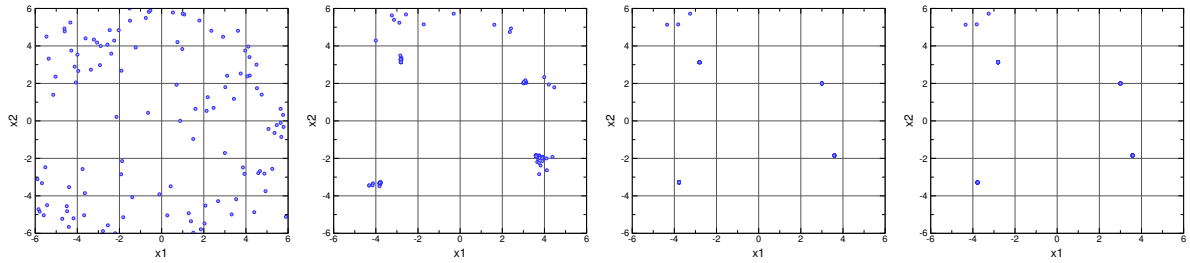


(m) fNCDE: 400 FEs (n) fNCDE: 4,000 FEs (o) fNCDE: 40,000 FEs (p) fNCDE: 400,000 FEs

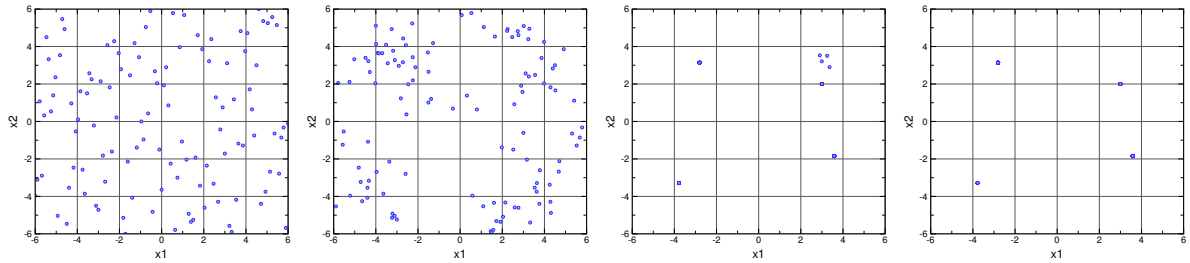
Figure 11: Convergence of individuals on function `F11`



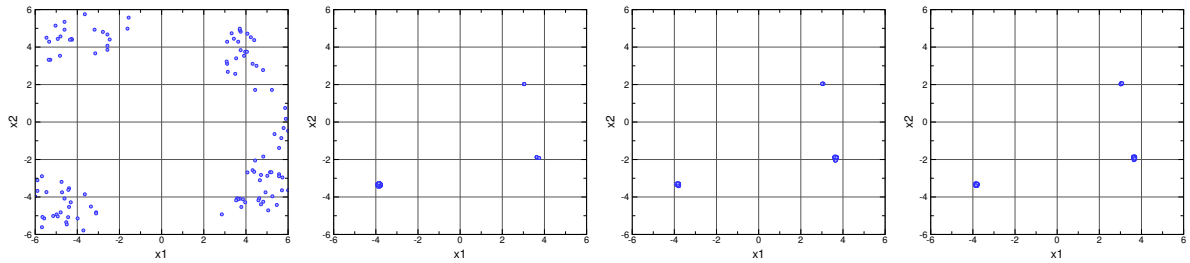
(a) fNRAND1: 400 FEs (b) fNRAND1: 4,000 FEs (c) fNRAND1: 40,000 FEs (d) fNRAND1: 400,000 FEs



(e) fNSDE: 400 FEs (f) fNSDE: 4,000 FEs (g) fNSDE: 40,000 FEs (h) fNSDE: 400,000 FEs



(i) fSDE: 400 FEs (j) fSDE: 4,000 FEs (k) fSDE: 40,000 FEs (l) fSDE: 400,000 FEs



(m) fSHDE: 400 FEs (n) fSHDE: 4,000 FEs (o) fSHDE: 40,000 FEs (p) fSHDE: 400,000 FEs

Figure 12: Convergence of individuals on function `F11`

In terms of the feasible performance of the algorithms, figure 13 gives the average FF of the algorithms on each function. First, it is clear that function `F12` has proven very difficult for a number of the algorithms. The nature of function `F12` is six very small feasible regions where the four global optima and two local optima are. Due to this very small region, no algorithm has had all individuals in a population converge to the global optima. Despite this, the FF is unity for a number of the algorithms for `F12` demonstrating that all individuals have located a feasible area, but it is not a global optimum.

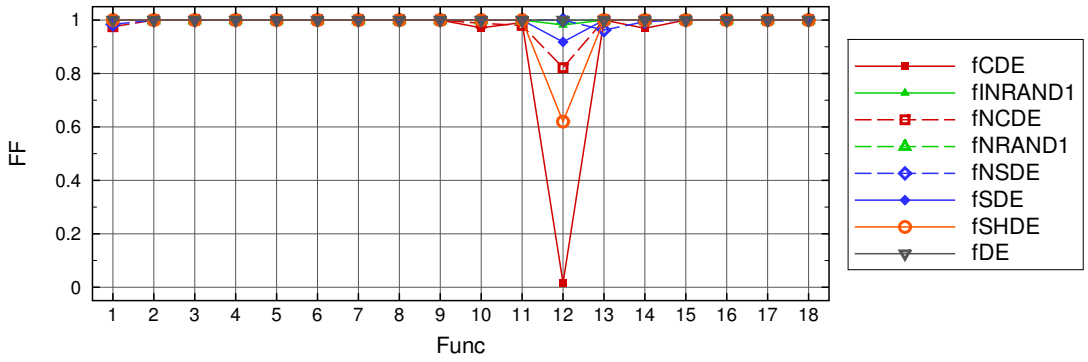
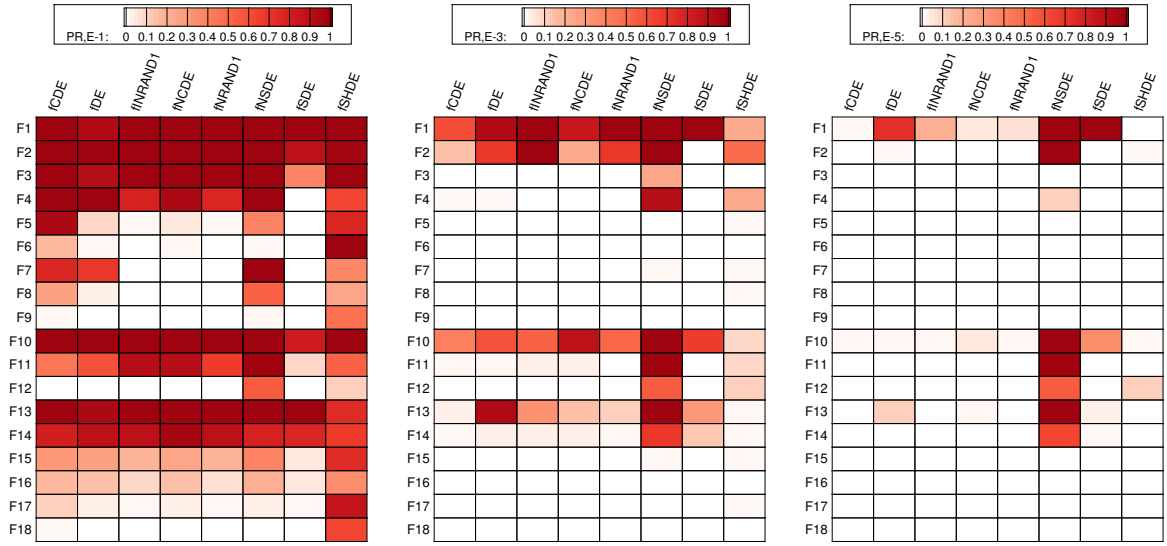


Figure 13: Feasible fraction for canonical DE-based algorithms run on low-dimensional suite with high FEs_{max}

6.2. Low FEs_{max}

While it is interesting to consider running the algorithms with a high number of function evaluations, some of the functions considered are low-dimensional and with few constraints so comparing the performance when a restricted number of function evaluations are allowed is an important investigation. In this work, the suggestion by Qu *et al.* [75] of an $\text{FEs}_{max} = 2000D\sqrt{N_g}$ is used. The population size remains as it was.

The peak ratio and success rates for all the algorithms tested on each function with the low allowable number of FEs_{max} are presented in tables S16 and S17. From this raw data, figure 14 provides an overview of the peak ratios at $\epsilon = 1 \times 10^{-1}, 1 \times 10^{-3}, 1 \times 10^{-5}$. Figure 15 gives the feasible fraction of each algorithm. It is interesting to note the difference in performance between the high and low allowable number of function evaluations. The fNCDE algorithm, that appears to perform best when the number of function evaluations is high, does not have the same level of performance when the function evaluations count is low, while the fNSDE algorithm, which performed well before (but not the best) performs very well here. As noted above, fNSDE has the tendency to very quickly form niches and locate the optima, but the niches are not stable and break down. However, when a restricted number of function evaluations is given, the niches do not have time to break down so the performance relative to the other algorithms is better. That being said, while the relative performance of fNSDE is better, the absolute performance compared to fNCDE with a high number of function evaluations is weaker, though this is to be expected.



(a) $\epsilon = 1 \times 10^{-1}$ (b) $\epsilon = 1 \times 10^{-3}$ (c) $\epsilon = 1 \times 10^{-5}$

Figure 14: Overview of peak ratios for canonical DE-based algorithms run on low-dimensional suite with low FEs$_{max}$

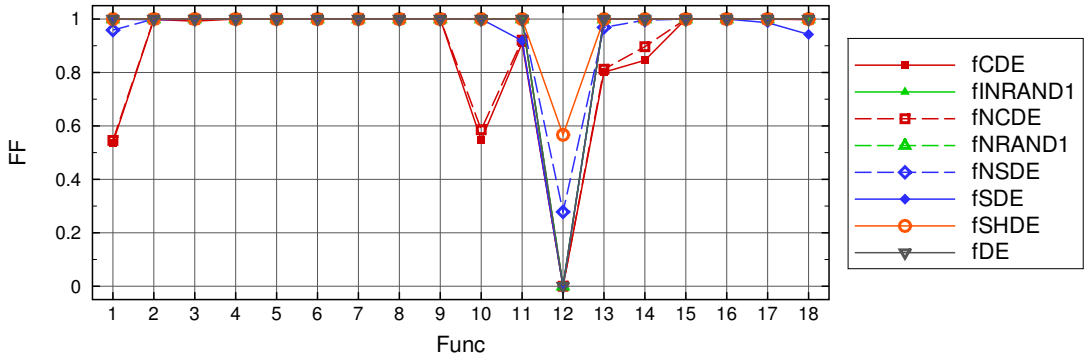


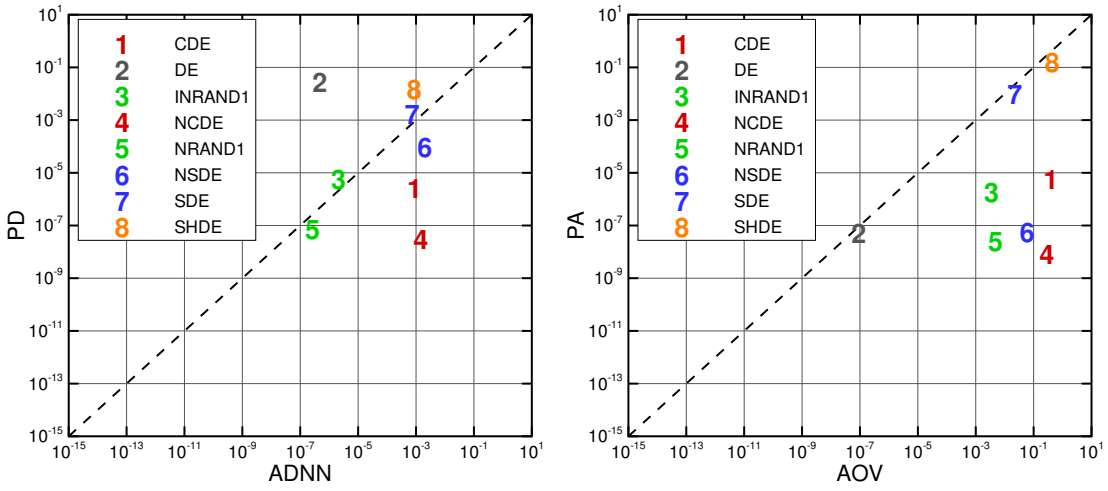
Figure 15: Feasible fraction for canonical DE-based algorithms run on low-dimensional suite with low FEs$_{max}$

6.3. Statistical Analysis

A final, statistical analysis is presented on the data to compare the performance of the algorithms against each other. Initially, analysis is presented that compares whether the algorithms have a bias towards clustering into niches, or converging onto the global optima. Finally, the peak ratio is one of the key performance metrics used in this work as it gives an indication of the relative performance of algorithms on standard test cases. As such, the peak ratio is used here to perform further statistical analysis.

Figure 16 plots the overall $ADNN$ against the overall PD, and the overall AOV against the overall PA for each algorithm. For each of these figures, the tendency for an algorithm to cluster is demonstrated if the results are towards the upper left quadrant, while if the results are towards the lower right quadrant then the algorithm has a tendency to converge onto the optima. If the results lie on a straight line then there is equal tendency to cluster and for those clusters to converge onto global optima. Plotting $ADNN$ against PD gives this bias in the design space, while plotting AOV against PA gives it in the objective space. In the design space, there is less bias between clustering and optima convergence, however, it is clear that fDE tends towards tightly clustering of the entire population into a single niche, but this comes at the expense of being far from most of the optima. On the other hand, fNCDE has much better

optima converging properties, but the niches tend to be less tightly clustered. In the objective space, the majority of the niching algorithms are able to converge onto objective values that are close to optimal (hence a low value for PA), but because of the niches formed, the AOV is not as low.



(a) Design space

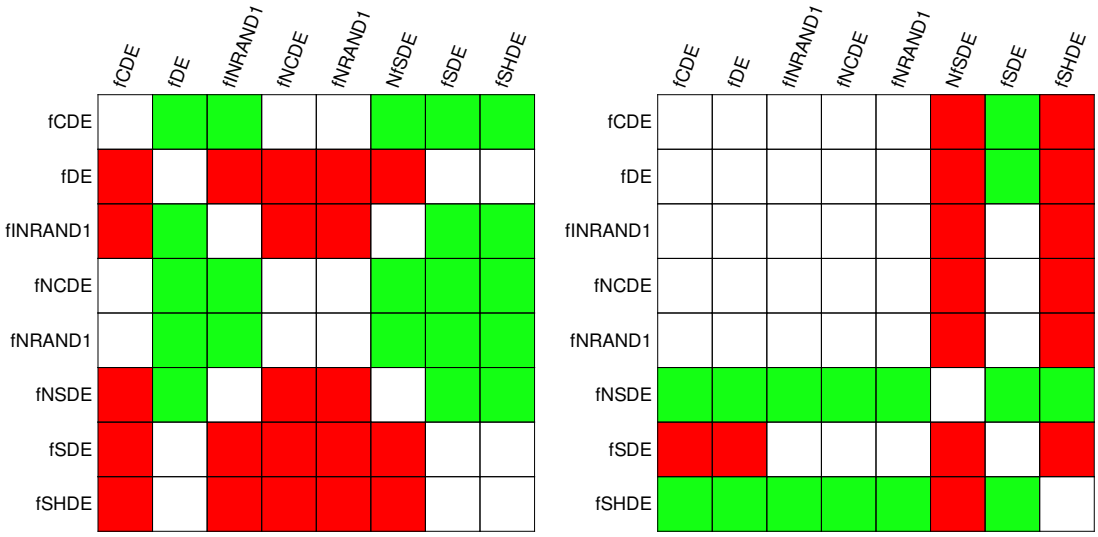
(b) Objective space

Figure 16: Representation of bias of algorithms in design (ADNN vs. PD) and objective spaces (AOV vs. PA)

To compare the performance of each algorithm against each other in terms of peak ratio, Wilcoxon rank-sum tests[73] are performed. The rank-sum test is used to test the null hypothesis that "*the median of a given metric of algorithms A and B are equal*" where the sample-set is the chosen metric of each function. The confidence level of the test is 95%. Both right-tailed and left-tailed p-values are calculated to test the alternate hypotheses of whether the median of A is greater than the median of B and whether the median of A is less than the median of B, respectively. When testing the peak ratios, if the median of A is greater than B, then A is said to have won, while if A is less than B then B has won, while if the null hypothesis is accepted then there is no difference at the confidence level.

Figure 17 gives the results of the rank-sum tests on the PR. A green box means that the algorithm on that row has a statistically different median and that the median is higher than the algorithm in that column (and therefore the row is determined to be better than the column algorithm), while a red box means that the medians are different and that the median of the algorithm of that row is lower than the algorithm of that column, so the row is determined to have lost. A white box means that the null hypothesis cannot be rejected at the 95% confidence level. Hence, each box is an independent rank-sum test with each sample size being the product of the number of functions tested and number of tolerances considered (90 in this case).

When the number of function evaluations is high, the comparison of PR leads to three distinct best performing algorithms, which are fCDE, fNCDE and fNRAND1. On the other hand, when the function evaluations are restricted, the clear winner (beating all other algorithms) is fNSDE. As noted before, fNSDE is excellent at quickly forming niches, but the longer the algorithm evolves for, the more these niches tend to break down. The simple to implement fNRAND1, and its less complex cousin, fINRAND1, have good overall performance, but the more complex algorithm wins against the less complex one. This is to be expected, but the trade-off in complexity is still important.



(a) High FEs_{max} (b) Low FEs_{max}

Figure 17: Result of rank-sum test on PR at all tolerance levels (green - row beats column, red- row loses to column, white - no statistical difference)

7. Investigation of DE Strategy

All of the results presented above use the canonical form of DE as a basis, with the *rand/1* mutation strategy. However, as noted in section 3, there are different forms that a DE algorithm can take, hence this is investigated here. In this section, the effect of mutation strategy and DE algorithm is considered on the low-dimensional function suite when combined with the various niching strategies.

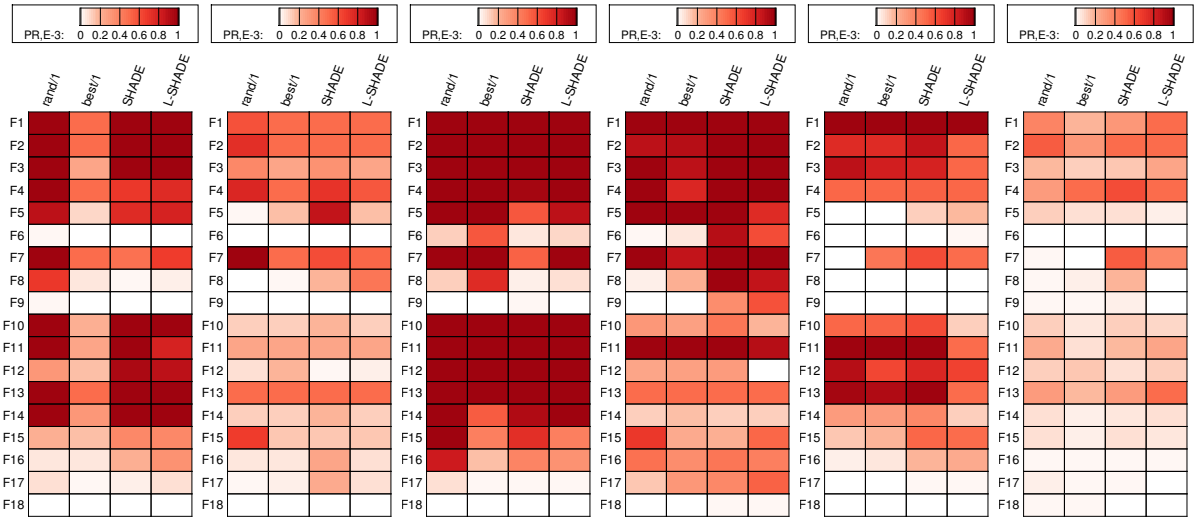
Four different DE strategies are compared. The rand/1 mutation strategy was used for the results presented above, so this acts as a baseline against which to compare other strategies. The best/1 mutation strategy is the second method used for comparison. All of the canonical DE-based algorithms presented in section 5 remain as they are except for changing the mutation strategy from equation 4 to equation 5. When determining the best individual in the population, the feasibility rules procedure is used, so if any feasible individuals exist, the one with the lowest objective function is the best, while if no feasible individuals exist, the one with the lowest constraint violation, ϕ, is the best. The final two strategies considered are SHADE and L-SHADE, when combined with the constrained niching approaches. In both of these algorithms, the p-best individuals are again determined by feasible rules. Finally, in L-SHADE the N^{min} parameter has to be appropriately set. When performing regular global optimization, N^{min} is set based on the minimum number of individuals for the particular mutation strategy. However, in multimodal optimization if $N^{min} < N_g$ then the algorithm will not be able to locate all of the optima. So in this work $N^{min} = N/2$ such that the benefits of the population reduction can still be exploited, but not at the expense of optima locating.

As before, two investigations are presented (high and low number of function evaluations) to consider the relative performance of all of the algorithms. All parameters are kept as before. Furthermore, every combination of DE strategy and constrained niching method are considered on the low-dimensional function suite.

Table 8 gives the OCS (with $\epsilon = 1 \times 10^{-4}$). In this sense, the more recent DE variants (SHADE and L-SHADE) appear to perform well. The rand/1 strategy is remarkably efficient, while the best/1 strategy is generally poorly performing. Peak ratio comparisons with high and low number of function evaluations are given in figures 18 and 19, respectively. As before, the two neighbourhood-based algorithms appear to perform relatively well with the number of function evaluations is high, and the exploitative nature of fNSDE that was seen before, still plays out when SHADE and L-SHADE strategies are used.

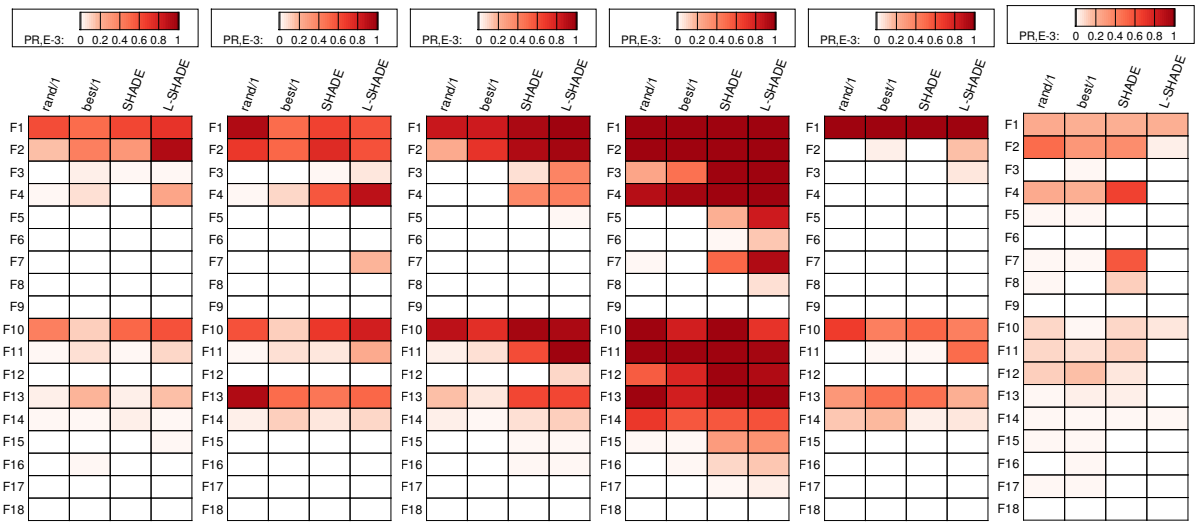
Table 8: OCS (with $\epsilon = 1 \times 10^{-4}$) for difference strategies of DE-based algorithms run on low-dimensional suite with high FEs$_{max}$

Strategy	fCDE	fDE	fNCDE	fNSDE	fSDE	fSHDE
rand/1	247504	297529	188973	208261	296498	358597
best/1	400000	399556	205170	229178	315513	397995
SHADE	250292	323059	235743	170723	301945	364892
L-SHADE	239104	255540	226542	134433	250602	366044



(a) fCDE (b) fDE (c) fNCDE (d) fNSDE (e) fSDE (f) fSHDE

Figure 18: Overview of peak ratios (at $\epsilon = 1 \times 10^{-3}$) for differing DE strategies run on low-dimensional suite with high FEs$_{max}$



(a) fCDE (b) fDE (c) fNCDE (d) fNSDE (e) fSDE (f) fSHDE

Figure 19: Overview of peak ratios (at $\epsilon = 1 \times 10^{-3}$) for differing DE strategies run on high-dimensional suite with low FEs$_{max}$

To validate the relative performance of all strategy/niching combinations, rank-sum tests are again performed on the peak ratios at high and low numbers of function evaluations. The results are given in figures 20 and 21. For the high number of function evaluations, across the board, fNCDE performs the best, with at least one mutation strategy/fNCDE combination beating all other constrained niching algorithms. SHADE and L-SHADE perform very well when combined with all mutation strategies. With the exception of the combination of rand/1 with fNCDE, the SHADE and L-SHADE strategies outperform rand/1 and best/1 for each of the niching methods. However, when the number of function evaluations is low, it is clear that fNSDE combines with either SHADE or L-SHADE is the clear best performing approach. It is possible that the performance of the L-SHADE approach could be improved further by tweaking the N^{min} for each individual problem if N_g is known *a-priori*. In this work, a black-box knowledge of the functions was assumed, so N^{min} was set relatively large.

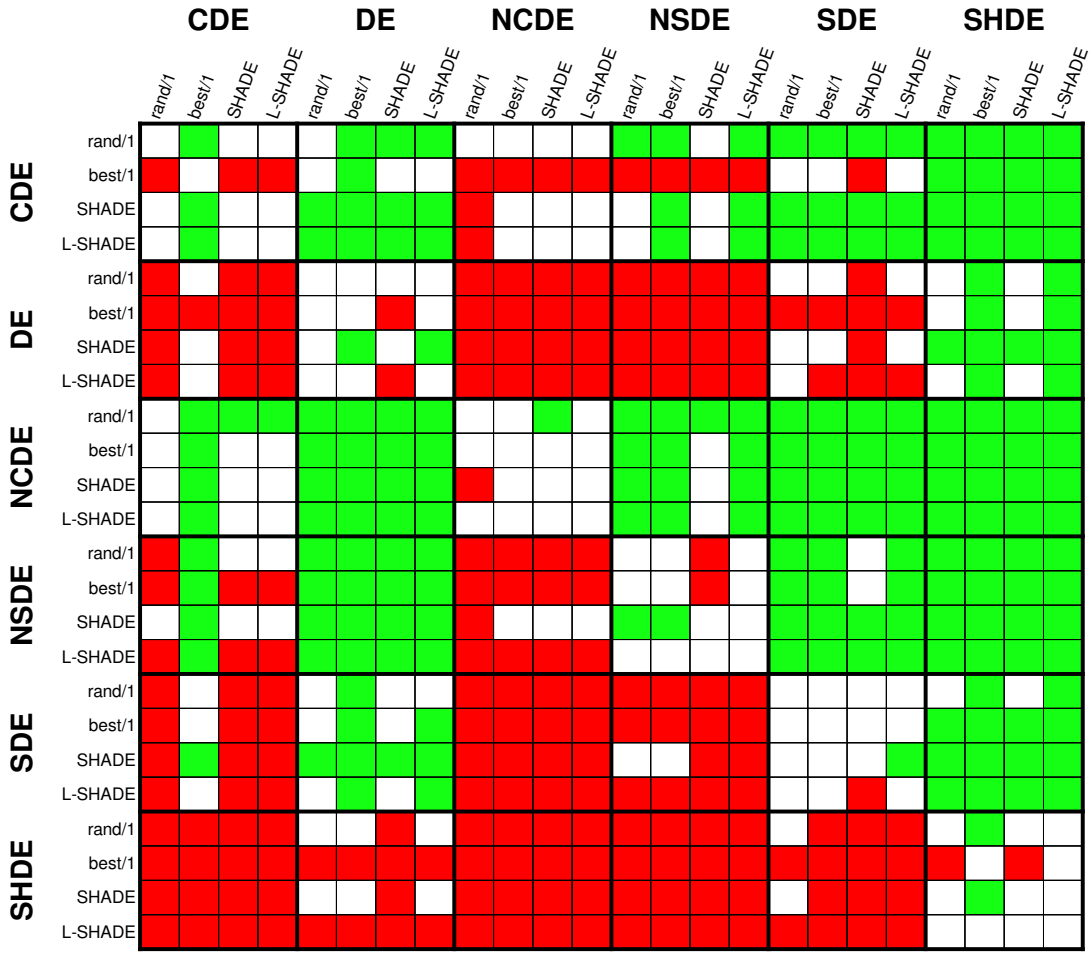


Figure 20: Results of rank-sum test on PR at all tolerance levels for differing DE strategies run on low-dimensional suite with high FEs$_{max}$ (green - row beats column, red- row loses to column, white - no statistical difference)

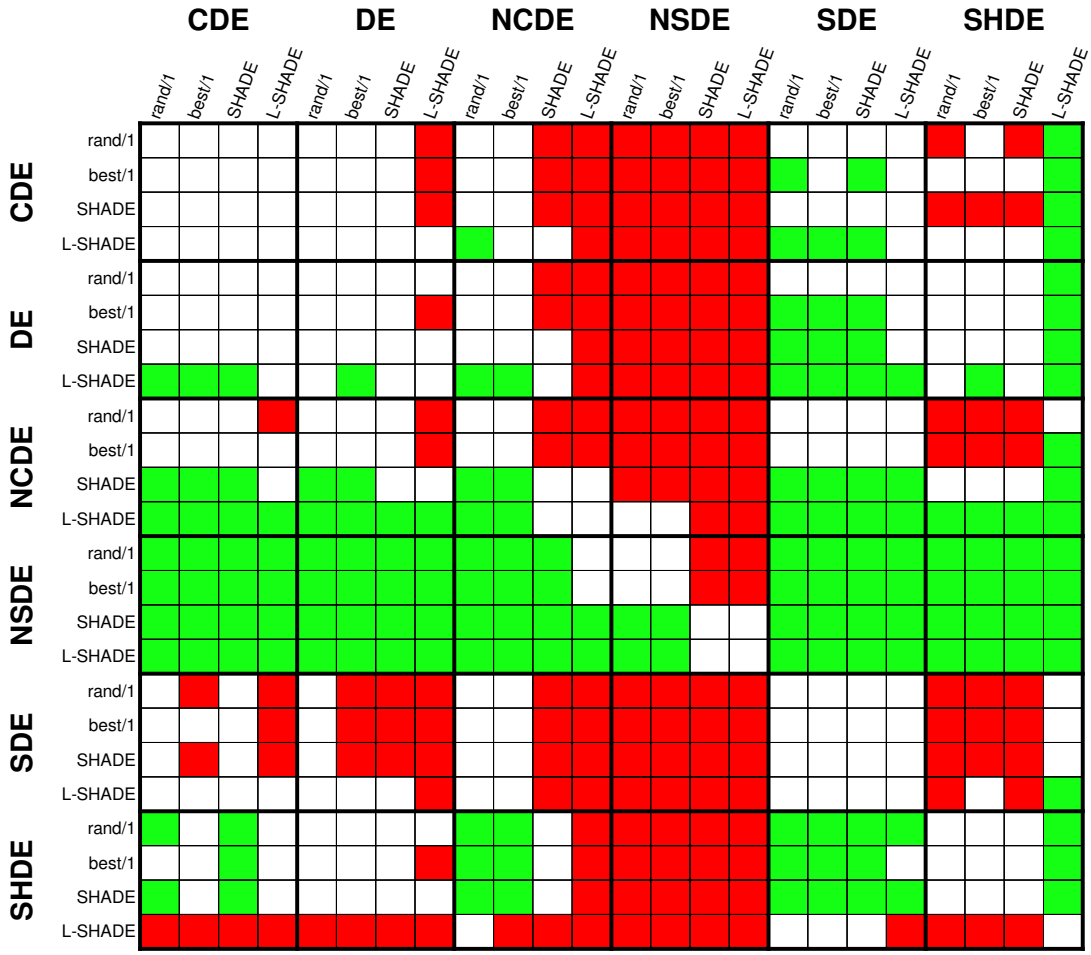


Figure 21: Results of rank-sum test on PR at all tolerance levels for differing DE strategies run on low-dimensional suite with low FEs$_{max}$ (green - row beats column, red- row loses to column, white - no statistical difference)

8. Comparison Against NSGA-II for Niching

The results of the various studies presented above demonstrate the effectiveness of using feasible selection for constraint handling with DE-based niching algorithms to solve constrained multimodal optimization problems. Although the relative performance of the algorithms is dependent on the pre-determined maximum number of function evaluations, the algorithms that were shown to consistently perform well were fNRAND1, fNCDE (with the rand/1 strategy), fNSDE (with the SHADE strategy) and fNSDE (with the L-SHADE strategy). As such, these are put forward as candidates to compare to the only other known constrained niching algorithm, which is that presented by Deb and Saha [27]. Furthermore, to enhance the comparison of the methods, the study presented in this section uses a single test problem that scales with dimensionality and constraints.

8.1. NSGA-II for Niching

The work of Deb and Saha [27], which is the only known previous work on niching in the presence of constraints, used a modified form of NSGA-II with binary tournament selection (as also used in this paper) for handling constraints. In that work, no name was given to the algorithm, hence for clarity, it is herein called feasible niching-NSGA-II (fNNSGA-II).

The approach employed in fNNSGA-II is to solve a bi-objective problem. Given the n-th individual in the current population, the objective vector is given as $\mathbf{f}(\mathbf{x}_n) = [f_1(\mathbf{x}_n), f_2(\mathbf{x}_n)]$. The problem objective of that individual is used directly as the first objective function, $f_1(\mathbf{x}_n) = f(\mathbf{x}_n)$, while the second objective function uses a metric that gives an indication of the characteristics of the objective function at an individual's location. Deb and Saha [27] suggested a number of methods for determining the second objective, with one being to use the local gradient (such that this would be zero at an optimum, then use the local Hessian to determine whether this was a minimum or a maximum). However, they recommended a Hooke-Jeeves exploratory search, and this is the method used here to determine the second objective.

For a given individual, the Hooke-Jeeves search proceeds by starting at that individual's location in the design space, evaluating the objective function at a small positive and negative perturbation in the first design variable and continuing along the direction that is best. This continues until all design variables have been considered. The second objective is the count of the number of solutions that are found to improve the objective function, and infeasible solutions are ignored. Hence, if an individual is at an optimum, then the second objective will have a value of zero. The disadvantage of this type of search is that to evaluate f_2 for a single individual in the population, $2D$ function evaluations are required.

The fNNSGA-II procedure then proceeds in a similar manner to the regular NSGA-II (see Deb *et al.* [28]), but with two modifications. The first is that the definition of domination is changed, such that in fNNSGA-II, solution $\mathbf{x}_a$ dominates solution $\mathbf{x}_b$ if $f_2(\mathbf{x}_a) < f_2(\mathbf{x}_b)$ and $f_1(\mathbf{x}_a) \leq f_1(\mathbf{x}_b)$. This allows solutions with equal f_2 values to have the same rank in the algorithm and therefore allows multiple solutions at minimum points. Second, a rank-degrading concept is introduced. This rank-degrading happens after non-dominated sorting has occurred and all individuals are assigned a domination rank. Individuals of equal rank are arranged in fronts. Rank-degrading proceeds by considering all individuals in the non-dominated front in ascending order of f_1 and assigns a large rank to any individuals in the entire population who have equal f_2, an f_1 that is within δ_f of the current individual in the front in question, and a Euclidean distance that is within δ_x of the current individual in the front in question. This continues through all individuals in all fronts until all individuals in the population are degraded or assigned a non-domination level. Finally, constraints are handled using binary tournament selection, as previously outlined in this paper. This is used to determine non-domination, as well as when selecting individuals for the pool. Algorithm 14 gives the overall fNNSGA-II algorithm. Simulated binary crossover and polynomial mutation [76] are used for the evolutionary operators.

Algorithm 14 fNNSGA-II algorithm

Randomly initialise individuals
Calculate f_1 and determine f_2 via Hooke-Jeeves search: algorithm 15
Non-dominated sorting and rank-degrading
Calculate crowding distance
while FEs<FEs$_{max}$ **do**
  Fill pool of candidate solutions from current population
  Generate test population from pool using crossover and mutation
  Calculate objectives f_1 and f_2 (algorithm 15) of test population
  Non-dominated sorting and rank-degrading of combined current and test population
  Calculate crowding distance of combined current and test population
  Determine new population based on rank, then crowding distance
end while

8.2. Test Problems

To perform the dimensionality and constraint number study, the test problem studied is the CMMP problem defined by Deb and Saha [26, 27]. The problem is defined in section 4.1 of this paper so only an outline of the dimensionality and number of constraints is given here. Design space dimensionality from 5 to 30 is considered, and each is considered with 1, 2, 3 and 4 constraints. A summary of this is provided in table 9 where the naming convention of CMMP follows that defined by Deb and Saha [26, 27].

Algorithm 15 fNNSGA-II Hooke-Jeeves search algorithm to calculate f_2

for $n = 1 \rightarrow N$ **do**
  Set $f_2(\mathbf{x}_n) = 0$ and $\boldsymbol{\zeta} = \mathbf{x}_n$
  for $d = 1 \rightarrow D$ **do**
    Set $\boldsymbol{\zeta}_+ = \boldsymbol{\zeta}_- = \boldsymbol{\zeta}$
    $\zeta_+^d = \zeta_+^d + \delta_{hj}$ and $\zeta_-^d = \zeta_-^d - \delta_{hj}$
    Calculate objective and constraints for $\boldsymbol{\zeta}_+$ and $\boldsymbol{\zeta}_-$
    if $f_1(\boldsymbol{\zeta}_+) < f_1(\mathbf{x}_n)$ and $\boldsymbol{\zeta}_+$ is feasible **then**
      $f_2(\mathbf{x}_n) + +$
    end if
    if $f_1(\boldsymbol{\zeta}_-) < f_1(\mathbf{x}_n)$ and $\boldsymbol{\zeta}_-$ is feasible **then**
      $f_2(\mathbf{x}_n) + +$
    end if
    Set $\boldsymbol{\zeta}$ to be the best of $\boldsymbol{\zeta}_+$, $\boldsymbol{\zeta}_-$ and $\boldsymbol{\zeta}$
  end for
end for

Table 9: Definition of problems for dimensionality study

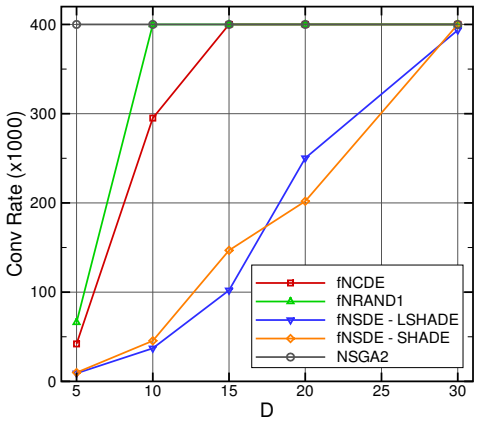
Name	D	p	N_g	$f(\mathbf{x}^*)$
CMMP(5,2,0)	5	1	2	1.0
CMMP(10,2,0)	10	1	2	1.0
CMMP(15,2,0)	15	1	2	1.0
CMMP(20,2,0)	20	1	2	1.0
CMMP(30,2,0)	30	1	2	1.0
CMMP(5,4,0)	5	2	4	1.354679802955665
CMMP(10,4,0)	10	2	4	1.189636052021373
CMMP(15,4,0)	15	2	4	1.128814769279581
CMMP(20,4,0)	20	2	4	1.097476180632552
CMMP(30,4,0)	30	2	4	1.065550775558326
CMMP(5,8,0)	5	3	8	1.729843561973526
CMMP(10,8,0)	10	3	8	1.393589488353573
CMMP(15,8,0)	15	3	8	1.265010184747948
CMMP(20,8,0)	20	3	8	1.199351958194447
CMMP(30,8,0)	30	3	8	1.133156192190737
CMMP(5,16,0)	5	4	16	2.064161725645909
CMMP(10,16,0)	10	4	16	1.608094265226821
CMMP(15,16,0)	15	4	16	1.407893201166005
CMMP(20,16,0)	20	4	16	1.305417931095781
CMMP(30,16,0)	30	4	16	1.202777020972990

All parameters in the algorithms presented in this paper are as they were defined in the previous studies. The parameters for fNNSGA-II are as defined in Deb and Saha [27], except that the population is kept the same as the other algorithms, $N = 40\sqrt{DN_g}$. As before, 50 independent runs of each algorithm on each function were performed.

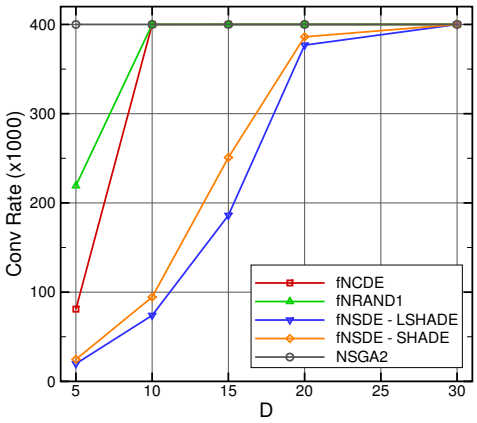
8.3. Results

A summary of the convergence rate for each of the algorithms tested is shown in figure 22. This is shown plotted against the number of dimensions, for the four different numbers of constraints. Figures 23 and 24 show the PR at $\epsilon = 1 \times 10^{-3}$ and $\epsilon = 1 \times 10^{-5}$, respectively. The first observation is that fNNSGA-II performs poorly against the other, more state-of-the-art algorithms. It should be noted that fNNSGA-II was validated against the results presented in Deb and Saha [27] so results given here are representative. The poor performance is attributed to the use of a

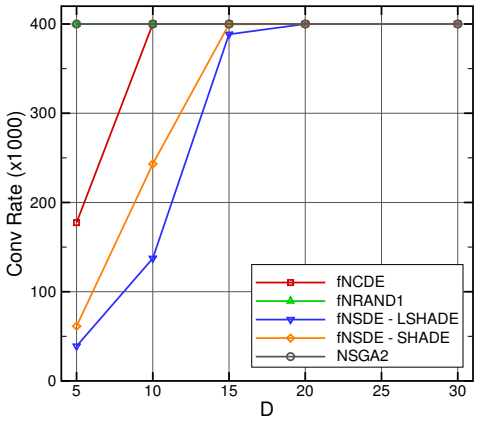
Hooke-Jeeves search to evaluate the second objective. This requires $2D$ function evaluations for each new individual so is expensive for anything but low-dimensional problems. On the other hand, the two fNSDE-based algorithms (which use SHADE and L-SHADE strategies) clearly perform the best of those tested. The L-SHADE strategy overall has lower convergence rates on all but one of the problems tested. It is particularly positive to note that when dimensionality is increased, the SHADE and L-SHADE strategies outperform the more classical DE strategies.



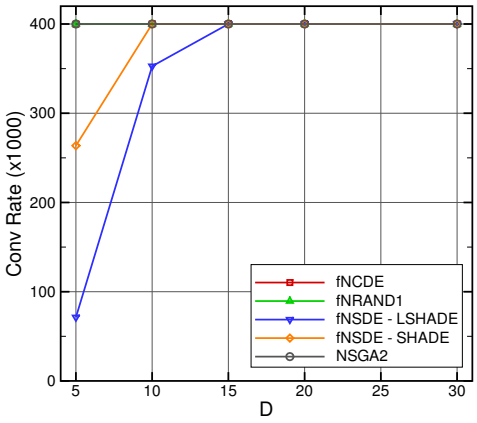
(a) 1 constraint



(b) 2 constraints

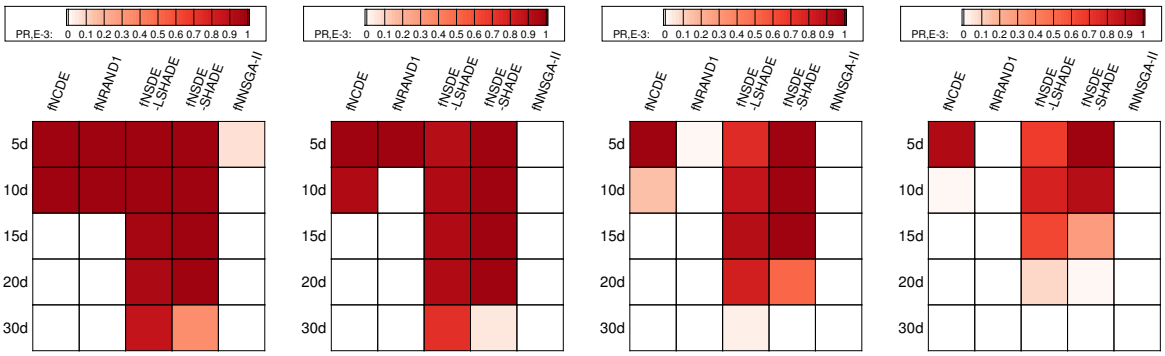


(c) 3 constraints



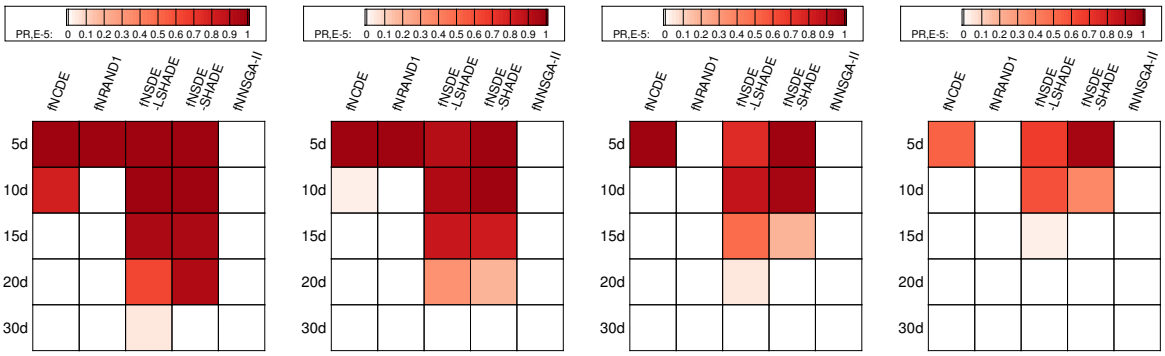
(d) 4 constraints

Figure 22: Convergence rate for algorithms in dimensionality study



(a) 1 constraint (b) 2 constraints (c) 3 constraints (d) 4 constraints

Figure 23: PR (at $\epsilon = 1 \times 10^{-3}$) of algorithms in dimensionality study



(a) 1 constraint (b) 2 constraints (c) 3 constraints (d) 4 constraints

Figure 24: PR (at $\epsilon = 1 \times 10^{-5}$) of algorithms in dimensionality study

Finally, a ranking is provided of the algorithms tested in the dimensionality study. Using a PR at $\epsilon = 1 \times 10^{-3}$, the ranking score, R for each of the algorithms is calculated by weighting the PR of an algorithm on a particular function by the product of the dimensionality and number of constraints of that function, and summing. Hence:

$$R = \sum_{i=1}^{\text{number of functions}} D_i p_i \text{PR}_i \tag{33}$$

This weighting emphasises performance of algorithms on more difficult (those with more constraints and higher dimensionality) functions.

Once R is calculated, it is used to determine a final rank. The R and final rankings of each algorithm is given in table 10. It is clear that in terms of peak ratio (and also, from figure 22, convergence rate) that fNSDE (with the L-SHADE strategy) is the best performing algorithm in the dimensionality study.

Table 10: Final ranking of algorithms in dimensionality study

	fNCDE	fNRAND1	fNSDE - LSHADE	fNSDE - SHADE	fNNSGA-II
R	81.7	25.1	430.8	357.6	0.4
Rank	3	4	1	2	5

9. Concluding Remarks

An investigation into constrained multimodal optimization using an evolutionary algorithm (differential evolution - DE) has been presented here. Numerous previous works have, independently, investigated comprehensively, constrained optimization using evolutionary algorithms and unconstrained multimodal optimization. However, little work to date has been presented on constrained multimodal optimization. As such, this paper has presented various constrained DE-based niching techniques that use a feasibility rules-domination selection procedure. A suite of 18 low-dimensional analytical constrained functions that contain multiple global minima has also been developed that contain nine test problems from the literature and nine new test problems. While this suite contains relatively simple problems, it is hoped that it can act as a basis for development of more complicated problems.

The constrained niching algorithms have been run on the 18 test problems with both a high and low pre-determined number of function calls. Results demonstrate that local neighbourhood-based crowding and species niching algorithms are generally highly effective at quickly forming local niches, but that neighbourhood-based species DE struggles to maintain those local niches. Hence, when the number of function calls is low, neighbourhood-based species performs well. Using a canonical DE often yields reasonable results, with local niches formed around global and local optima quickly, but often the niches are unstable and canonical DE tends to converge to a single niche. An investigation of the DE strategy also reveals that using more recent DE variants (such as SHADE and L-SHADE) have the advantage of removing the need to tune mutation and crossover parameters, while also often improving the performance of niching algorithms. This is particularly prevalent when low numbers of function calls are permitted. Finally, a dimensionality study was performed which also compared high-performing algorithms to a niching form of NSGA-II. This demonstrated that the NSGA-II method, which uses a Hooke-Jeeves search to determine a second objective for NSGA-II, uses too many function evaluations for high-dimensional search problems. On the other hand, using L-SHADE with neighbourhood-based species niching, provides excellent results.

This study is one of very few studies into constrained multimodal optimization and as such there are a substantial number of unanswered questions regarding this field. For example, the effect of using other nature-inspired algorithms should be investigated and also other niching techniques when combined with these algorithms. Also, there are many constraint handling techniques so studies into the effect these have when combined with the niching techniques are invaluable. Further development into high-dimensional problems is also an area that in general has little investigation in the unconstrained multimodal optimization literature, but this is also important for constrained multimodal optimization.

Data Access Statement

The data required to support the conclusions of this paper are presented in the paper. The source code of the best performing algorithm in this paper (fNSDE using L-SHADE) and the source code of the analytical function suite is available for download from the University of Bristol data repository, `data.bris`, at `https://doi.org/10.5523/bris.2lnc51etw0obe2s2lml7n2nciq`.

Detail of Supplementary Material

Supplementary material is provided with the paper containing raw data that is linked to in the paper. The supplementary material is:

- Tables S1 to S13: details of the parameter tuning process
- Tables S14 and S15: peak ratios and success ratios using a high number of function evaluations
- Tables S16 and S17: peak ratios and success ratios using a low number of function evaluations

Acknowledgements

This work was carried out using the computational facilities of the Advanced Computing Research Centre, University of Bristol; `https://www.acrc.bris.ac.uk/`.

References

[1] C. B. Allen and T. C. S. Rendall. Computational-fluid-dynamics-based optimisation of hovering rotors using radial basis functions for shape parameterisation and mesh deformation. *Optimization and Engineering*, 14:97–118, 2013.

[2] G. K. W. Kenway and J. R. R. A. Martins. Multi-point high-fidelity aerostructural optimization of a transport aircraft configuration. *Journal of Aircraft*, 51(1):144–160, 2014.

[3] A. Mishra, D. Mavriplis, and J. Sitaraman. Time-dependent aeroelastic adjoint-based aerodynamic shape optimization of helicopter rotors in forward flight. *AIAA Journal*, 54(12):3813–3827, 2016.

[4] T. D. Economon, F. Palacios, S. R. Copeland, T. W. Lukaczyk, and J. J. Alonso. SU2: An open-source suite for multiphysics simulation and design. *AIAA Journal*, 54(3):828–846, 2016.

[5] D. J. Poole, C. B. Allen, and T. C. S. Rendall. High-fidelity aerodynamic shape optimization using efficient orthogonal modal design variables with a constrained global optimizer. *Computers & Fluids*, 143:1–15, 2017.

[6] J. H. Holland. *Adaptation in Natural and Artificial Systems*. The University of Michigan Press, 1975.

[7] R. Storn and K. Price. Differential evolution a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 11:341–359, 1997.

[8] S. Das, S. S. Mullick, and P. N. Suganthan. Recent advances in differential evolution - an updated survey. *Swarm and Evolutionary Computation*, 27:1–30, 2016.

[9] G. Venter and J. Sobieszczanski-Sobieski. Particle swarm optimization. *AIAA Journal*, 41(8):1583–1589, 2003.

[10] H. J. C. Barbosa and A. C. C. Lemonge. A new adaptive penalty scheme for genetic algorithms. *Information Sciences*, 156(3-4):215–251, 2003.

[11] C. A. Coello Coello. Use of a self-adaptive penalty approach for engineering optimization problems. *Computers in Industry*, 41(2):113–127, 2000.

[12] J. J. Liang and P. N. Suganthan. Dynamic multi-swarm particle swarm optimizer with a novel constraint-handling mechanism. In *2006 IEEE Congress on Evolutionary Computation*, Vancouver, Canada, 2006.

[13] D. J. Poole, C. B. Allen, and T. C. S. Rendall. A generic framework for handling constraints with agent-based optimization algorithms and application to aerodynamic design. *Optimization and Engineering*, 18(3):659–691, 2017.

[14] K. Deb. An efficient constraint handling method for genetic algorithms. *Computer Methods in Applied Mechanics and Engineering*, 186:311–338, 2000.

[15] G. Toscano Pulido and C. A. Coello Coello. A constraint handling mechanism for particle swarm optimization. In *2004 IEEE Congress on Evolutionary Computation*, Portland, Oregon, 2004.

[16] C. L. Sun, J. C. Zeng, and J. S. Pan. An improved vector particle swarm optimization for constrained optimization problems. *Information Sciences*, 181:1153–1163, 2011.

[17] K. A. De Jong. *Analysis of the behavior of a class of genetic adaptive systems*. PhD thesis, University of Michigan, 1975.

[18] R. Thomsen. Multimodal optimization using crowding-based differential evolution. In *Congress on Evolutionary Computation*, Portland, Oregon, 2004.

[19] D. E. Goldberg and J. Richardson. Genetic algorithms with sharing for multimodal function optimization. In *Proceedings of the Second International Conference on Genetic Algorithms on Genetic algorithms and their application*, 1987.

[20] A. Petrowski. A clearing procedure as a niching method for genetic algorithms. In *Proceedings of IEEE International Conference on Evolutionary Computation*, Nagoya, Japan, 1996.

[21] J. P. Li, M. E. Balazs, G. T. Parks, and P. J. Clarkson. A species conserving genetic algorithm for multimodal function optimization. *Evolutionary Computation*, 10(3):207–234, 2002.

[22] B. Y. Qu, P. N. Suganthan, and J. J. Liang. Differential evolution with neighborhood mutation for multimodal optimization. *IEEE Transactions on Evolutionary Computation*, 16(5):601–614, 2012.

[23] Y.-H. Zhang, Y.-J. Gong, H.-X. Zhang, T.-L. Gu, and J. Zhang. Toward fast niching evolutionary algorithms: A locality sensitive hashing-based approach. *IEEE Transactions on Evolutionary Computation*, 21(3):347–362, 2017.

[24] E. Mezura-Montes and C. A. Coello Coello. Constraint handling in nature-inspired numerical optimization: Past, present and future. *Swarm and Evolutionary Computation*, 1:173–194, 2011.

[25] X. Li, M. G. Epitropakis, K. Deb, and A. Engelbrecht. Seeking multiple solutions: An updated survey on niching methods and their applications. *IEEE Transactions on Evolutionary Computation*, 21(4):518–538, 2017.

[26] K. Deb and A. Saha. Finding multiple solutions for multimodal optimization problems using a multi-objective evolutionary approach. In *GECCO '10 Proceedings of the 12th annual conference on genetic and evolutionary computation*, Portland, Oregon, 2010.

[27] K. Deb and A. Saha. Multimodal optimization using a bi-objective evolutionary algorithm. *Evolutionary Computation*, 20(1):27–62, 2012.

[28] K. Deb. A fast and elitist multiobjective genetic algorithm: NSGA-II. *IEEE Transactions on Evolutionary Computation*, 6(2):182–197, 2002.

[29] D. J. Poole, C. B. Allen, and T. C. S. Rendall. Global optimization of wing aerodynamic optimization case exhibiting multimodality. *Journal of Aircraft*, 2018. Published online.

[30] K. Deb and S. Gulati. Design of truss-structures for minimum weight using genetic algorithms. *Finite Elements in Analysis and Design*, 37(5):447–465, 2001.

[31] R. Datta and K. Deb, editors. *Evolutionary Constrained Optimization*. Springer India, 2015.

[32] X. Hu and R. Eberhart. Solving constrained nonlinear optimization problems with particle swarm optimization. In *6th World Multiconference on Systemics, Cybernetics and Informatics (SCI 2002)*, Orlando, Florida, 2002.

[33] G. Coath and S. K. Halgamuge. A comparison of constraint-handling methods for the application of particle swarm optimization to constrained nonlinear optimization problems. In *2003 IEEE Congress on Evolutionary Computation*, Canberra, Australia, 2003.

[34] H. J. C. Barbosa, A. C. C. Lemonge, and H. S. Bernardino. A critical review of adaptive penalty techniques in evolutionary computation. In R. Datta and K. Deb, editors, *Evolutionary Constrained Optimization*. Springer India, 2015.

[35] K. E. Parsopoulos and M. N. Vrahatis. Particle swarm optimization method for constrained optimization problems. In *Euro-International Symposium on Computational Intelligence 2002*, pages 214–220, 2002.
[36] H. Lu and W. Chen. Dynamic-objective particle swarm optimization for constrained optimization problems. *Journal of Combinatorial Optimization*, 12(4):409–419, 2006.
[37] K. Zielinski and R. Laur. Constrained single-objective optimization using particle swarm optimization. In *2006 IEEE Congress on Evolutionary Computation*, Vancouver, Canada, 2006.
[38] T. Takahama and S. Sakai. Constrained optimization by the α-constrained particle swarm optimizer. *Journal of Advanced Computational Intelligence and Intelligent Informatics*, 9(3):282–289, 2005.
[39] T. Takahama and S. Sakai. Constrained optimization by the ϵ-constrained particle swarm optimizer with epsilon-level control. *Advances in Soft Computing*, 29:1019–1029, 2005.
[40] T. Takahama and S. Sakai. Constrained optimization by the ϵ-constrained differential evolution with gradient-based mutation and feasible elites. In *2006 IEEE Congress on Evolutionary Computation*, Vancouver, Canada, 2006.
[41] G. Venter and R. T. Haftka. Constrained particle swarm optimization using a bi-objective formulation. *Structural Multidisciplinary Optimization*, 40:65–76, 2010.
[42] Y. Wang and Cai. Z. Combining multiobjective optimization with differential evolution to solve constrained optimization problems. *IEEE Transactions on Evolutionary Computation*, 16(1):117–134, 2012.
[43] G. N. Vanderplaats. *Numerical Optimization Techniques for Engineering Design*. Vanderplaats Research and Development Inc., 3rd edition, 1999.
[44] T. Miyashita. Particle swarm optimization using projection matrix for behaviour constraints. In *11th AIAA/ISSMO Multidisciplinary Analysis and Optimization Conference*, Portsmouth, Virginia, 2006. AIAA Paper 2006-6911.
[45] H. Lu and W. Chen. Self-adaptive velocity particle swarm optimization for solving constrained optimization problems. *Journal of Global Optimization*, 41(3):427–445, 2008.
[46] Z. Li, J. J. Liang, and Z. Shang. Differential evolution with dynamic constraint-handling mechanism. In *2010 IEEE Congress on Evolutionary Computation*, Barcelona, Spain, 2010.
[47] E. Mezura-Montes, J. Velazquez-Reyes, and C. A. Coello Coello. Modified differential evolution for constrained optimization. In *2006 IEEE Congress on Evolutionary Computation*, Vancouver, Canada, 2006.
[48] A. W. Mohamed and H. Z. Sabry. Constrained optimization based on modified differential evolution algorithm. *Information Sciences*, 194:171–208, 2012.
[49] S. M. Elsayed, R. A. Sarker, and D. L. Essam. An improved self-adaptive differential evolution algorithm for optimization problems. *IEEE Transactions on Industrial Informatics*, 9(1):89–99, 2013.
[50] J. Brest, B. Boskovic, and V. Zumer. An improved self-adaptive differential evolution algorithm in single objective constrained real-parameter optimization. In *2010 IEEE Congress on Evolutionary Computation (CEC)*, Barcelona, Spain, 2010.
[51] R. Mallipeddi and P. N. Suganthan. Ensemble of constraint handling techniques. *IEEE Transactions on Evolutionary Computation*, 14(4), 2010.
[52] E. Mezura-Montes, M. E. Miranda-Varela, and R. d. C. Gomez-Ramon. Differential evolution in constrained numerical optimization: An empirical study. *Information Sciences*, 180(22):4223–4262, 2010.
[53] S. Das, S. Maity, B-Y. Qu, and P. N. Suganthan. Real-parameter evolutionary multimodal optimization – a survey of the state-of-the-art. *Swarm and Evolutionary Computation*, 1(2):71–88, 2011.
[54] S. W. Mahfoud. *Niching Methods for Genetic Algorithms*. PhD thesis, University of Illinois at Urbana Champaign, 1995.
[55] D. E. Goldberg, K. Deb, and J. Horn. Massive multimodality, deception, and genetic algorithms. In R. Manner and B. Manderick, editors, *Parallel Problem Solving from Nature, 2: Proceedings of the Second Conference on Parallel Problem Solving from Nature*, pages 37–46, Brussels, Belgium, 1992.
[56] X. Li. Efficient differential evolution using speciation for multimodal function optimization. In *GECCO '05 Proceedings of the 7th annual conference on Genetic and evolutionary computation*, Washington, D.C., 2005.
[57] M. G. Epitropakis, X. Li, and E. K. Burke. A dynamic archive niching differential evolution algorithm for multimodal optimization. In *2013 IEEE Congress on Evolutionary Computation*, Cancun, Mexico, 2013.
[58] S. Biswas, S. Kundu, and S. Das. Inducing niching behavior in differential evolution through local information sharing. *IEEE Transactions on Evolutionary Computation*, 19(2):246–263, 2015.
[59] M. G. Epitropakis, V. P. Plagianakos, and M. N. Vrahatis. Finding multiple global optima exploiting differential evolution's niching capability. In *2011 IEEE Symposium on Differential Evolution (SDE)*, Paris, France, 2011.
[60] M. G. Epitropakis, V. P. Plagianakos, and M. N. Vrahatis. Multimodal optimization using niching differential evolution with index-based neighborhoods. In *2012 IEEE Congress on Evolutionary Computation*, Brisbane, Australia, 2012.
[61] E. L. Yu and P. N. Suganthan. Ensemble of niching algorithms. *Information Sciences*, 180(15):2815–2833, 2010.
[62] R. Storn and K. Price. Differential evolution - a simple and efficient adaptive scheme for global optimization over continuous spaces. Technical report, ICSI, UC Berkeley, 1995. TR-95-012.
[63] S. Das and P. N. Suganthan. Differential evolution: A survey of the state-of-the-art. *IEEE Transactions on Evolutionary Computation*, 15(1):4–31, 2011.
[64] F. Neri and V. Tirronen. Recent advances in differential evolution: a survey and experimental analysis. *Artificial Intelligence Review*, 33:61–106, 2010.
[65] R. Tanabe and A. S. Fukunaga. Success-history based parameter adaptation for differential evolution. In *2013 IEEE Congress on Evolutionary Computation (CEC)*, Cancun, Mexico, 2013.
[66] R. Tanabe and A. S. Fukunaga. Improving the search performance of shade using linear population size reduction. In *2014 IEEE Congress on Evolutionary Computation (CEC)*, Beijing, China, 2014.
[67] J. Zhang and A. C. Sanderson. JADE: Adaptive differential evolution with optional external archive. *IEEE Transactions on Evolutionary Computation*, 13(5):945–958, 2009.

[68] X. Li, A. Engelbrecht, and M. G. Epitropakis. Benchmark functions for cec'2013 special session and competition on niching methods for multimodal function optimization. Technical report, Evolutionary Computation and Machine Learning Group, RMIT University, Australia, 2013.

[69] A. Saha and K. Deb. *Simulated Evolution and Learning. SEAL 2010. Lecture Notes in Computer Science, vol 6457*, chapter A Bi-criterion Approach to Multimodal Optimization: Self-adaptive Approach. Springer, Berlin, Heidelberg, 2010.

[70] J. Mwaura, A. P. Engelbrecht, and F. V. Nepocumeno. Performance measures for niching algorithms. In *2016 IEEE Congress on Evolutionary Computation*, Vancouver, Canada, 2016.

[71] E. Mezura-Montes, C. A. Coello Coello, and E. I. Tun-Morales. Simple feasibility rules and differential evolution for constrained optimization. In *MICAI 2004: Advances in Artificial Intelligence*. Springer, Berlin, 2004.

[72] W.-J. Yu, J.-Y. Ji, Y.-J. Gong, Q. Yang, and J. Zhang. A tri-objective differential evolution approach for multimodal optimization. *Information Sciences*, 423:1–23, 2018.

[73] F. Wilcoxon. Individual comparisons by ranking methods. *Biometrics Bulletin*, 1(6):80–83, 1945.

[74] A. P. Piotrowski, M. J. Napiorkowski, J. J. Napiorkowski, and P. M. Rowinski. Swarm intelligence and evolutionary algorithms: Performance versus speed. *Information Sciences*, 384:34–85, 2017.

[75] B. Y. Qu, J. J. Liang, Z. Y. Wang, Q. Chen, and P. N. Suganthan. Novel benchmark functions for continuous multimodal optimization with comparative results. *Swarm and Evolutionary Computation*, 26:23, 2016.

[76] K. Deb and R. B. Agrawal. Simulated binary crossover for continuous search space. *Complex Systems*, 9(2):115–148, 1995.

Table S1: Results of F and CR tuning of fCDE

F	CR	Wins	Losses	OCS	Rank
0.1	0.1	1	0	247,505	1
0.9	0.1	1	0	283,688	3
0.1	0.9	1	0	253,977	2
0.9	0.9	0	3	295,153	4

Table S2: Results of F and CR tuning of fDE

F	CR	Wins	Losses	OCS	Rank
0.1	0.1	1	0	348,985	2
0.9	0.1	2	0	297,529	1
0.1	0.9	0	3	395,576	4
0.9	0.9	1	1	307,942	3

Table S3: Results of F and CR tuning of fINRAND1

F	CR	Wins	Losses	OCS	Rank
0.1	0.1	0	0	249,180	4
0.9	0.1	0	0	246,445	3
0.1	0.9	0	0	240,289	2
0.9	0.9	0	0	233,775	1

Table S4: Results of F and CR tuning of fNRAND1

F	CR	Wins	Losses	OCS	Rank
0.1	0.1	0	0	242,184	4
0.9	0.1	0	0	235,580	3
0.1	0.9	0	0	235,555	2
0.9	0.9	0	0	213,234	1

Table S5: Results of F and CR tuning of fNCDE ($m = 10$)

F	CR	Wins	Losses	OCS	Rank
0.1	0.1	0	2	217,922	3
0.9	0.1	0	2	229,426	4
0.1	0.9	2	0	193,958	2
0.9	0.9	2	0	188,973	1

Table S6: Results of F and CR tuning of fNSDE ($m = 10$)

F	CR	Wins	Losses	OCS	Rank
0.1	0.1	1	2	351,827	3
0.9	0.1	2	0	208,261	1
0.1	0.9	0	3	398,694	4
0.9	0.9	2	0	214,398	2

Table S7: Results of F and CR tuning of fSDE ($m = 10$, $\sigma = 1\%$)

F	CR	Wins	Losses	OCS	Rank
0.1	0.1	0	0	332,702	3
0.9	0.1	0	0	298,087	2
0.1	0.9	0	0	333,297	4
0.9	0.9	0	0	279,759	1

Table S8: Results of F and CR tuning of fSHDE ($\sigma = 1\%$)

F	CR	Wins	Losses	OCS	Rank
0.1	0.1	0	0	375,866	1
0.9	0.1	0	0	381,539	3
0.1	0.9	0	0	376,797	2
0.9	0.9	0	0	381,639	4

Table S9: Results of m tuning of fNCDE ($F = 0.9$, $CR = 0.9$)

m	Wins	Losses	OCS	Rank
5	0	0	199,764	2
10	1	0	188,973	1
20	0	1	218,097	3

Table S10: Results of σ tuning of fSDE ($F = 0.1$, $CR = 0.1$, $m = 20$)

σ	Wins	Losses	OCS	Rank
0.1%	0	1	358,621	3
1%	0	0	324,040	2
10%	1	0	296,498	1

Table S11: Results of m tuning of fSDE ($F = 0.9$,$CR = 0.9$,$\sigma = 10\%$)

m	Wins	Losses	OCS	Rank
5	0	0	316,497	3
10	0	0	308,632	2
20	0	0	296,498	1

Table S12: Results of m tuning of fNSDE ($F = 0.9$, $CR = 0.1$)

m	Wins	Losses	OCS	Rank
5	0	1	264571	3
10	1	0	208261	1
20	0	0	257985	2

Table S13: Results of σ tuning of fSHDE ($F = 0.1$, $CR = 0.1$)

σ	Wins	Losses	OCS	Rank
0.1%	1	0	358,597	1
1%	0	0	375,866	3
10%	0	0	352,453	2

Table S14: Overall Peak Ratios and Success Rates for canonical DE-based algorithms run on functions F1 to F9 with high FEs_{max}

		fCDE		fDE		fINRAND1		fNCDE		fNRAND1		fNSDE		fSDE		fSHDE	
	ϵ	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR
F1	1E-1	1.000	1.000	0.590	0.180	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
	1E-2	1.000	1.000	0.590	0.180	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.750	0.540
	1E-3	1.000	1.000	0.590	0.180	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.390	0.100
	1E-4	1.000	1.000	0.590	0.180	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.310	0.000
	1E-5	1.000	1.000	0.590	0.180	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.260	0.000
F2	1E-1	1.000	1.000	0.720	0.440	1.000	1.000	1.000	1.000	1.000	1.000	0.890	0.780	0.770	0.540	1.000	1.000
	1E-2	1.000	1.000	0.720	0.440	1.000	1.000	1.000	1.000	1.000	1.000	0.890	0.780	0.750	0.500	1.000	1.000
	1E-3	1.000	1.000	0.720	0.440	1.000	1.000	1.000	1.000	1.000	1.000	0.890	0.780	0.750	0.500	0.550	0.220
	1E-4	1.000	1.000	0.720	0.440	1.000	1.000	1.000	1.000	1.000	1.000	0.890	0.780	0.740	0.480	0.420	0.020
	1E-5	1.000	1.000	0.720	0.440	1.000	1.000	1.000	1.000	1.000	1.000	0.890	0.780	0.740	0.480	0.310	0.000
F3	1E-1	1.000	1.000	0.380	0.060	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
	1E-2	1.000	1.000	0.380	0.060	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.980	0.920	0.315	0.000
	1E-3	1.000	1.000	0.380	0.060	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.890	0.580	0.170	0.000
	1E-4	1.000	1.000	0.380	0.060	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.815	0.420	0.015	0.000
	1E-5	1.000	1.000	0.380	0.060	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.760	0.260	0.000	0.000
F4	1E-1	1.000	1.000	0.770	0.540	1.000	1.000	1.000	1.000	1.000	1.000	0.980	0.960	0.680	0.360	0.570	0.140
	1E-2	1.000	1.000	0.770	0.540	1.000	1.000	1.000	1.000	1.000	1.000	0.980	0.960	0.530	0.060	0.500	0.140
	1E-3	1.000	1.000	0.770	0.540	1.000	1.000	1.000	1.000	1.000	1.000	0.980	0.960	0.510	0.020	0.290	0.080
	1E-4	0.990	0.980	0.770	0.540	1.000	1.000	1.000	1.000	1.000	1.000	0.980	0.960	0.510	0.020	0.140	0.000
	1E-5	0.970	0.960	0.770	0.540	1.000	1.000	1.000	1.000	1.000	1.000	0.980	0.960	0.510	0.020	0.010	0.000
F5	1E-1	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.500	0.020	0.775	0.600
	1E-2	1.000	1.000	0.797	0.180	0.998	0.980	1.000	1.000	1.000	1.000	1.000	1.000	0.000	0.000	0.667	0.400
	1E-3	0.895	0.880	0.003	0.000	0.005	0.000	1.000	1.000	0.013	0.000	1.000	1.000	0.000	0.000	0.120	0.020
	1E-4	0.497	0.480	0.000	0.000	0.000	0.000	1.000	1.000	0.000	0.000	0.940	0.580	0.000	0.000	0.010	0.000
	1E-5	0.245	0.220	0.000	0.000	0.000	0.000	1.000	1.000	0.000	0.000	0.573	0.040	0.000	0.000	0.000	0.000
F6	1E-1	1.000	1.000	0.175	0.000	0.014	0.000	0.968	0.280	0.022	0.000	0.913	0.020	0.000	0.000	1.000	1.000
	1E-2	0.974	0.780	0.000	0.000	0.000	0.000	0.764	0.000	0.000	0.000	0.388	0.000	0.000	0.000	0.041	0.000
	1E-3	0.003	0.000	0.000	0.000	0.000	0.000	0.114	0.000	0.000	0.000	0.007	0.000	0.000	0.000	0.000	0.000
	1E-4	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
	1E-5	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
F7	1E-1	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.000	0.000	0.310	0.000
	1E-2	1.000	1.000	0.990	0.980	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.000	0.000	0.070	0.000
	1E-3	1.000	1.000	0.990	0.980	0.930	0.880	1.000	1.000	0.990	0.980	1.000	1.000	0.000	0.000	0.020	0.000
	1E-4	0.970	0.960	0.990	0.980	0.000	0.000	1.000	1.000	0.010	0.000	1.000	1.000	0.000	0.000	0.000	0.000
	1E-5	0.630	0.440	0.760	0.640	0.000	0.000	0.810	0.680	0.000	0.000	1.000	1.000	0.000	0.000	0.000	0.000
F8	1E-1	1.000	1.000	1.000	1.000	0.060	0.000	1.000	1.000	0.135	0.000	1.000	1.000	0.000	0.000	0.203	0.020
	1E-2	1.000	1.000	0.080	0.000	0.000	0.000	0.713	0.100	0.000	0.000	0.733	0.100	0.000	0.000	0.133	0.020
	1E-3	0.682	0.520	0.000	0.000	0.000	0.000	0.105	0.000	0.000	0.000	0.033	0.000	0.000	0.000	0.003	0.000
	1E-4	0.007	0.000	0.000	0.000	0.000	0.000	0.010	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
	1E-5	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
F9	1E-1	1.000	1.000	0.132	0.000	0.001	0.000	0.250	0.000	0.000	0.000	0.541	0.000	0.000	0.000	0.376	0.040
	1E-2	0.437	0.100	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.322	0.040
	1E-3	0.001	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.008	0.000
	1E-4	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
	1E-5	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000

Table S15: Overall Peak Ratios and Success Rates for canonical DE-based algorithms run on functions F10 to F18 with high FEs$_{max}$

		fCDE		fDE		fINRAND1		fNCDE		fNRAND1		fNSDE		fSDE		fSHDE	
	ϵ	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR
F10	1E-1	1.000	1.000	0.120	0.000	0.954	0.580	1.000	1.000	1.000	1.000	0.304	0.000	0.532	0.000	0.990	0.920
	1E-2	1.000	1.000	0.120	0.000	0.954	0.580	1.000	1.000	1.000	1.000	0.304	0.000	0.532	0.000	0.342	0.000
	1E-3	1.000	1.000	0.120	0.000	0.954	0.580	1.000	1.000	1.000	1.000	0.304	0.000	0.520	0.000	0.116	0.000
	1E-4	1.000	1.000	0.120	0.000	0.954	0.580	1.000	1.000	1.000	1.000	0.304	0.000	0.464	0.000	0.070	0.000
	1E-5	1.000	1.000	0.120	0.000	0.950	0.540	1.000	1.000	1.000	1.000	0.304	0.000	0.438	0.000	0.066	0.000
F11	1E-1	1.000	1.000	0.250	0.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.740	0.340
	1E-2	1.000	1.000	0.250	0.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.445	0.000
	1E-3	1.000	1.000	0.250	0.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.990	0.960	0.225	0.000
	1E-4	1.000	1.000	0.250	0.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.975	0.900	0.180	0.000
	1E-5	1.000	1.000	0.250	0.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.950	0.800	0.080	0.000
F12	1E-1	0.310	0.000	0.065	0.000	0.935	0.740	1.000	1.000	1.000	1.000	0.260	0.000	0.915	0.700	0.110	0.000
	1E-2	0.310	0.000	0.065	0.000	0.935	0.740	1.000	1.000	1.000	1.000	0.260	0.000	0.915	0.700	0.110	0.000
	1E-3	0.310	0.000	0.065	0.000	0.935	0.740	1.000	1.000	1.000	1.000	0.260	0.000	0.915	0.700	0.110	0.000
	1E-4	0.310	0.000	0.065	0.000	0.935	0.740	1.000	1.000	1.000	1.000	0.260	0.000	0.915	0.700	0.110	0.000
	1E-5	0.310	0.000	0.065	0.000	0.935	0.740	1.000	1.000	1.000	1.000	0.260	0.000	0.915	0.700	0.110	0.000
F13	1E-1	1.000	1.000	0.500	0.000	1.000	1.000	1.000	1.000	1.000	1.000	0.500	0.000	1.000	1.000	0.880	0.760
	1E-2	1.000	1.000	0.500	0.000	1.000	1.000	1.000	1.000	1.000	1.000	0.500	0.000	1.000	1.000	0.410	0.100
	1E-3	1.000	1.000	0.500	0.000	1.000	1.000	1.000	1.000	1.000	1.000	0.500	0.000	0.970	0.940	0.280	0.000
	1E-4	1.000	1.000	0.500	0.000	1.000	1.000	1.000	1.000	1.000	1.000	0.500	0.000	0.960	0.920	0.250	0.000
	1E-5	1.000	1.000	0.500	0.000	1.000	1.000	1.000	1.000	1.000	1.000	0.500	0.000	0.950	0.900	0.050	0.000
F14	1E-1	1.000	1.000	0.102	0.000	0.994	0.940	1.000	1.000	1.000	1.000	0.110	0.000	0.354	0.000	0.664	0.080
	1E-2	1.000	1.000	0.102	0.000	0.942	0.480	0.994	0.940	1.000	1.000	0.110	0.000	0.314	0.000	0.188	0.000
	1E-3	1.000	1.000	0.102	0.000	0.940	0.460	0.994	0.940	1.000	1.000	0.110	0.000	0.286	0.000	0.076	0.000
	1E-4	0.978	0.800	0.102	0.000	0.936	0.420	0.994	0.940	1.000	1.000	0.110	0.000	0.270	0.000	0.052	0.000
	1E-5	0.858	0.180	0.102	0.000	0.934	0.420	0.994	0.940	1.000	1.000	0.110	0.000	0.264	0.000	0.014	0.000
F15	1E-1	1.000	1.000	0.675	0.280	0.970	0.780	1.000	1.000	0.998	0.980	0.713	0.040	0.932	0.580	0.775	0.040
	1E-2	0.988	0.900	0.675	0.280	0.970	0.780	0.998	0.980	0.998	0.980	0.705	0.040	0.660	0.060	0.138	0.000
	1E-3	0.203	0.000	0.675	0.280	0.970	0.780	0.993	0.960	0.998	0.980	0.698	0.040	0.128	0.000	0.075	0.000
	1E-4	0.010	0.000	0.585	0.000	0.968	0.760	0.955	0.700	0.998	0.980	0.682	0.040	0.050	0.000	0.003	0.000
	1E-5	0.000	0.000	0.333	0.000	0.958	0.720	0.915	0.520	0.963	0.760	0.680	0.040	0.037	0.000	0.000	0.000
F16	1E-1	1.000	1.000	0.996	0.900	0.976	0.680	1.000	1.000	1.000	1.000	0.601	0.000	0.818	0.020	0.447	0.000
	1E-2	0.662	0.000	0.618	0.000	0.833	0.000	0.977	0.640	0.843	0.000	0.469	0.000	0.222	0.000	0.053	0.000
	1E-3	0.050	0.000	0.043	0.000	0.426	0.000	0.821	0.020	0.103	0.000	0.467	0.000	0.038	0.000	0.017	0.000
	1E-4	0.000	0.000	0.002	0.000	0.070	0.000	0.511	0.000	0.001	0.000	0.462	0.000	0.015	0.000	0.003	0.000
	1E-5	0.000	0.000	0.000	0.000	0.007	0.000	0.277	0.000	0.000	0.000	0.459	0.000	0.008	0.000	0.000	0.000
F17	1E-1	0.985	0.760	0.998	0.960	0.966	0.660	0.938	0.420	0.922	0.260	0.559	0.000	0.087	0.000	0.985	0.780
	1E-2	0.787	0.300	0.185	0.000	0.169	0.000	0.448	0.000	0.046	0.000	0.347	0.000	0.003	0.000	0.120	0.000
	1E-3	0.066	0.000	0.004	0.000	0.001	0.000	0.065	0.000	0.000	0.000	0.136	0.000	0.000	0.000	0.035	0.000
	1E-4	0.003	0.000	0.000	0.000	0.000	0.000	0.007	0.000	0.000	0.000	0.015	0.000	0.000	0.000	0.000	0.000
	1E-5	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
F18	1E-1	0.263	0.000	0.017	0.000	0.000	0.000	0.003	0.000	0.000	0.000	0.010	0.000	0.000	0.000	0.978	0.560
	1E-2	0.013	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.088	0.000
	1E-3	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.001	0.000
	1E-4	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
	1E-5	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000

Table S16: Overall Peak Ratios and Success Rates for canonical DE-based algorithms run on functions F1 to F9 with low FEs$_{max}$

		fCDE		fDE		fINRAND1		fNCDE		fNRAND1		fNSDE		fSDE		fSHDE	
	ϵ	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR
F1	1E-1	1.000	1.000	0.930	0.860	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000
	1E-2	1.000	1.000	0.930	0.860	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.540	0.200
	1E-3	0.610	0.380	0.930	0.860	1.000	1.000	0.850	0.720	1.000	1.000	1.000	1.000	1.000	1.000	0.230	0.040
	1E-4	0.070	0.000	0.930	0.860	0.850	0.720	0.330	0.140	0.350	0.120	1.000	1.000	1.000	1.000	0.050	0.000
	1E-5	0.020	0.000	0.730	0.520	0.210	0.080	0.050	0.000	0.080	0.000	1.000	1.000	1.000	1.000	0.000	0.000
F2	1E-1	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.890	0.800	0.970	0.940
	1E-2	0.920	0.840	1.000	1.000	1.000	1.000	0.910	0.820	1.000	1.000	1.000	1.000	0.110	0.000	0.770	0.540
	1E-3	0.160	0.040	0.690	0.540	0.980	0.960	0.230	0.060	0.690	0.440	1.000	1.000	0.000	0.000	0.500	0.140
	1E-4	0.010	0.000	0.090	0.020	0.260	0.080	0.010	0.000	0.040	0.000	1.000	1.000	0.000	0.000	0.110	0.000
	1E-5	0.000	0.000	0.010	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	1.000	0.000	0.000	0.010	0.000
F3	1E-1	0.995	0.980	0.915	0.720	1.000	1.000	0.995	0.980	0.995	0.980	0.995	0.980	0.395	0.020	1.000	1.000
	1E-2	0.090	0.000	0.050	0.000	0.135	0.000	0.065	0.000	0.090	0.000	0.870	0.540	0.005	0.000	0.295	0.000
	1E-3	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.260	0.000	0.000	0.000	0.000	0.000
	1E-4	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.020	0.000	0.000	0.000	0.000	0.000
	1E-5	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
F4	1E-1	1.000	1.000	1.000	1.000	0.790	0.620	0.950	0.900	0.770	0.600	1.000	1.000	0.000	0.000	0.630	0.260
	1E-2	0.590	0.380	0.390	0.160	0.000	0.000	0.060	0.020	0.000	0.000	1.000	1.000	0.000	0.000	0.430	0.120
	1E-3	0.020	0.000	0.010	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.910	0.820	0.000	0.000	0.240	0.020
	1E-4	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.630	0.440	0.000	0.000	0.000	0.000
	1E-5	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.120	0.000	0.000	0.000	0.000	0.000
F5	1E-1	0.948	0.800	0.098	0.000	0.005	0.000	0.048	0.000	0.005	0.000	0.383	0.000	0.000	0.000	0.762	0.480
	1E-2	0.010	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.520	0.160
	1E-3	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.010	0.000
	1E-4	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
	1E-5	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
F6	1E-1	0.172	0.000	0.001	0.000	0.000	0.000	0.001	0.000	0.000	0.000	0.003	0.000	0.000	0.000	1.000	1.000
	1E-2	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.021	0.000
	1E-3	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
	1E-4	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
	1E-5	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
F7	1E-1	0.780	0.620	0.680	0.540	0.000	0.000	0.000	0.000	0.000	0.000	1.000	1.000	0.000	0.000	0.370	0.100
	1E-2	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.570	0.340	0.000	0.000	0.050	0.000
	1E-3	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.010	0.000	0.000	0.000	0.020	0.000
	1E-4	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
	1E-5	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
F8	1E-1	0.265	0.020	0.022	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.527	0.020	0.000	0.000	0.258	0.020
	1E-2	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.138	0.000
	1E-3	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.003	0.000
	1E-4	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
	1E-5	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
F9	1E-1	0.015	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.008	0.000	0.000	0.000	0.471	0.020
	1E-2	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.186	0.000
	1E-3	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
	1E-4	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
	1E-5	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000

Table S17: Overall Peak Ratios and Success Rates for canonical DE-based algorithms run on functions F10 to F18 with low FEs$_{max}$

		fCDE		fDE		fINRAND1		fNCDE		fNRAND1		fNSDE		fSDE		fSHDE	
	ϵ	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR	PR	SR
F10	1E-1	1.000	1.000	0.998	0.980	0.998	0.980	1.000	1.000	1.000	1.000	0.998	0.980	0.826	0.160	0.988	0.880
	1E-2	0.998	0.980	0.984	0.840	0.976	0.760	1.000	1.000	0.996	0.960	0.996	0.960	0.766	0.100	0.330	0.000
	1E-3	0.414	0.000	0.586	0.000	0.534	0.000	0.894	0.360	0.506	0.000	0.994	0.940	0.664	0.000	0.088	0.000
	1E-4	0.042	0.000	0.098	0.000	0.064	0.000	0.278	0.000	0.080	0.000	0.994	0.940	0.588	0.000	0.022	0.000
	1E-5	0.006	0.000	0.016	0.000	0.006	0.000	0.048	0.000	0.010	0.000	0.992	0.940	0.352	0.000	0.008	0.000
F11	1E-1	0.450	0.080	0.585	0.080	0.905	0.680	0.905	0.680	0.670	0.180	1.000	1.000	0.085	0.000	0.535	0.100
	1E-2	0.045	0.000	0.080	0.000	0.240	0.000	0.225	0.000	0.130	0.000	1.000	1.000	0.010	0.000	0.280	0.000
	1E-3	0.005	0.000	0.005	0.000	0.025	0.000	0.040	0.000	0.000	0.000	1.000	1.000	0.000	0.000	0.085	0.000
	1E-4	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	1.000	1.000	0.000	0.000	0.010	0.000
	1E-5	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.985	0.940	0.000	0.000	0.000	0.000
F12	1E-1	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.550	0.020	0.000	0.000	0.105	0.000
	1E-2	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.550	0.020	0.000	0.000	0.105	0.000
	1E-3	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.550	0.020	0.000	0.000	0.105	0.000
	1E-4	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.550	0.020	0.000	0.000	0.105	0.000
	1E-5	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.550	0.020	0.000	0.000	0.105	0.000
F13	1E-1	1.000	1.000	0.960	0.920	1.000	1.000	1.000	1.000	1.000	1.000	1.000	1.000	0.980	0.960	0.750	0.520
	1E-2	0.460	0.240	0.960	0.920	0.930	0.860	0.630	0.360	0.700	0.500	1.000	1.000	0.640	0.320	0.280	0.060
	1E-3	0.040	0.000	0.940	0.880	0.330	0.120	0.140	0.000	0.110	0.020	1.000	1.000	0.300	0.020	0.020	0.000
	1E-4	0.000	0.000	0.580	0.340	0.050	0.000	0.010	0.000	0.020	0.000	1.000	1.000	0.130	0.000	0.000	0.000
	1E-5	0.000	0.000	0.110	0.000	0.000	0.000	0.010	0.000	0.000	0.000	1.000	1.000	0.030	0.000	0.000	0.000
F14	1E-1	0.810	0.180	0.900	0.400	0.886	0.320	0.970	0.760	0.894	0.420	0.794	0.020	0.774	0.080	0.662	0.020
	1E-2	0.122	0.000	0.220	0.000	0.162	0.000	0.300	0.000	0.176	0.000	0.706	0.000	0.448	0.000	0.110	0.000
	1E-3	0.018	0.000	0.030	0.000	0.024	0.000	0.032	0.000	0.016	0.000	0.694	0.000	0.126	0.000	0.012	0.000
	1E-4	0.002	0.000	0.004	0.000	0.000	0.000	0.000	0.000	0.002	0.000	0.686	0.000	0.012	0.000	0.000	0.000
	1E-5	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.632	0.000	0.002	0.000	0.000	0.000
F15	1E-1	0.302	0.000	0.275	0.000	0.188	0.000	0.260	0.000	0.193	0.000	0.398	0.000	0.043	0.000	0.755	0.160
	1E-2	0.013	0.000	0.018	0.000	0.003	0.000	0.005	0.000	0.005	0.000	0.068	0.000	0.000	0.000	0.063	0.000
	1E-3	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.005	0.000	0.000	0.000	0.005	0.000
	1E-4	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
	1E-5	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
F16	1E-1	0.168	0.000	0.156	0.000	0.092	0.000	0.150	0.000	0.079	0.000	0.212	0.000	0.043	0.000	0.347	0.000
	1E-2	0.008	0.000	0.006	0.000	0.002	0.000	0.005	0.000	0.002	0.000	0.025	0.000	0.003	0.000	0.020	0.000
	1E-3	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
	1E-4	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
	1E-5	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
F17	1E-1	0.114	0.000	0.026	0.000	0.013	0.000	0.031	0.000	0.006	0.000	0.035	0.000	0.003	0.000	0.873	0.220
	1E-2	0.001	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.001	0.000	0.000	0.000	0.061	0.000
	1E-3	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.001	0.000
	1E-4	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
	1E-5	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
F18	1E-1	0.007	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.622	0.000
	1E-2	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.005	0.000
	1E-3	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
	1E-4	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000
	1E-5	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000	0.000